NASA CR-137472

HYBRID 3-D
ROCKET TRAJECTORY PROGRAM

Prepared for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

WALLOPS STATION

WALLOPS ISLAND, VIRGINIA 23337

CONTRACT NO. NAS6-2369

WORK ORDER 7

(NASA-CR-137472) HYBRID 3-D ROCKET TRAJECTORY PROGRAM. PART 1: FORMULATION AND ANALYSIS. PART 2: COMPUTER (Computer Sciences Corp., Wallops Island, Va.) 112 p HC $8.75 CSCL 22C G3/30

N74-34296

Unclas 50856

CSC

COMPUTER SCIENCES CORPORATION

Hybrid 3-D

Rocket Trajectory Program

Part One: Formulation and Analysis

by

Louis C. P. Huang

Part Two: Computer Programming and User's Instruction

by

Ronald A. Cook

Prepared for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

WALLOPS STATION

WALLOPS ISLAND, VIRGINIA 23337

1973

Part One

Formulation and Analysis

A. INTRODUCTION

A rocket in flight has three linear and three angular degrees of freedom. A mathematical model m be devised which describes the motions within the six degrees of freedom ana this '6-D' model may be programmed for solution by a digital computer. Given a set of initial conditions and tables of rocket characteristics, the equations which make up the model may be evaluated repeatedly with respect to time to give an accurate simulation of the flight of a rocket.

The '6-D' model, however, has two deficiencies:

1. Significant effort is required in preparing the input data to be used by the computer program. A specimen '6-D' simulation shows that over 700 input data items are required. When trajectories must be predicted for a new type of rocket, only part of this data set may be available from the manufacturer. The remaining items must be calculated from aerodynamic theory. All the data must then be transcribed in the exact form required. As a result, preparation of the input data usually takes at least two man-weeks.

2. Each '6-D' computer run may require as much as four hours of expensive computer time.

For these reasons, models utilizing various sub-sets of the six degrees of freedom are used in trajectory simulation. A '3-D' model with only linear degrees of freedom is especially attractive, since the coefficients for the angular degrees of freedom are the most difficult to determine and the angular equations are the most time consuming for the computer to evaluate.

Of course, the '3-D' model is less accurate than the '6-D' model. This is because the model lacks angular motions and the thrust vector orientation is assumed to be aligned with the velocity vector. Unless the angle of attack is zero, this is not true.

Figure 1 shows a typical angle of attack versus time history of an unguided rocket. The time scale is divided into three periods. In Period I the rocket has a finite angle of attack and is said to be untrimmed. In Period II the angle of attack has been trimmed out to zero and in Period III the rocket is again untrimmed.

The manner in which angle of attack is generated is shown in Figure 2, a vector diagram of the forces on the rocket immediately after launch. Angle BOC is the launch elevation angle. OC is the thrust minus drag vector, aligned with the principal body axis. Vector CD represents the acceleration of gravity. Vector OD is the sum of OC (the thrust minus drag) and CD (gravity). The rocket is accelerated from rest in direction OD, but it is pointing in direction OC. Angle DOC represents the angle of attack.

This angle of attack is gradually reduced to zero by the stabilizing moment (Mo) produced by the rocket fins. Trimming does not take place immediately because the stabilizing tendency is resisted by the large moment of inertia about the rocket's principal axis.

If the angle of attack is assumed to be zero during Period I (i.e., before the rocket trims out), large errors in the predicted trajectory may result. This is the case when standard three linear degree of freedom simulations are used. Further errors may result from assumptions about other initial and launch conditions. To reduce these errors, the computer program described in this paper uses three separate subsections to predict trajectories. A launch rail subsection is used until the rocket has left its launcher. The program then switches to a special '3-D' section which computes motions in two linear and one angular degrees of freedom. This permits accurate simulation of Period I flight when the angle of attach is finite. When the rocket trims out, the program switches to the standard, three linear degrees of freedom model. This model is used throughout Periods II and III.

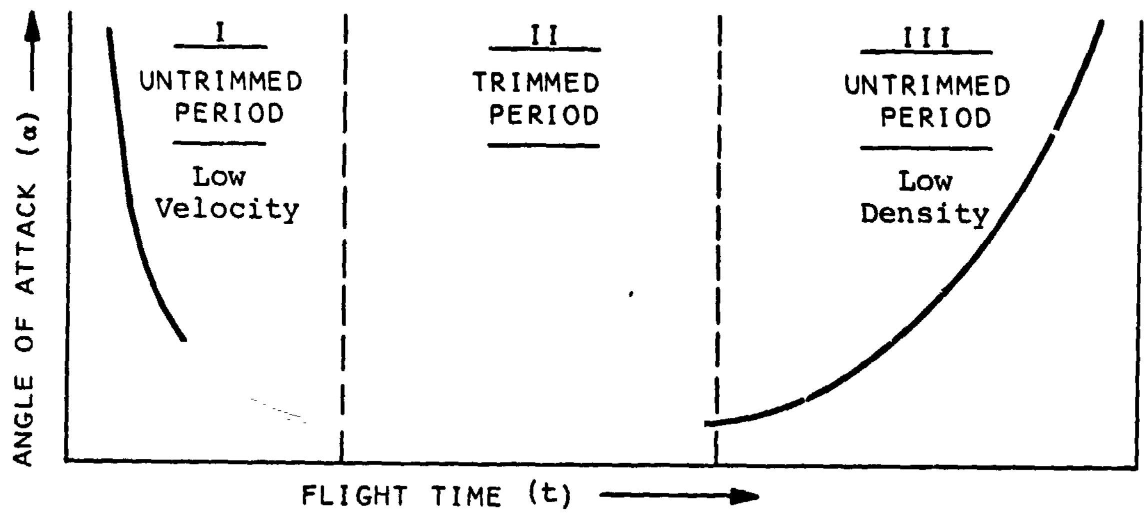
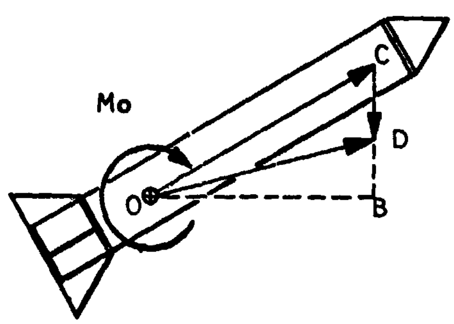


FIGURE 1: Angle-of-Attack/Time History



VECTOR		FORCE
OC	=	Thrust-Drag
+CD	=	Gravity
OD	=	Resultant

FIGURE 2: Forces on Rocket immediately after Launch

The standard '3-D' model can be used accurately in Period III despite the untrimmed condition because the rocket is not thrusting. This untrimmed condition results from the lack of restoring moment due to low atmospheric density. The aerodynamic force on the fins varies as the product of velocity squared and atmospheric density. Thus, although the velocity is high in Period III, the density is low enough that the restoring moment is insufficient to maintain a zero angle of attack.

In the absence of thrust, therefore, the untrimmed flight of Period III has no effect on the trajectory. Should there be an upper stage to be fired in Period III, the "HYBRID 3-D" program has a thrust vector control option to handle this case. There is also an option to change the aerodynamic drag on the spent rocket during this period.

By using the launch rail subsection, the special '3-D' section, the standard '3-D' section, and the thrust vector control and aerodynamic drag options, the program can calculate the rocket's impact point with sufficient accuracy for range safety hazard estimation and for the planning of payload recovery operations.

B. BASIC ASSUMPTIONS

The mathematical model used in the "HYBRID 3-D" program includes such effects as the variation of aerodynamic coefficients with Mach number, the change of engine thrust with time, and the change of rocket mass due to propellant burning and stage separation. But some simplifying assumptions have been necessary to speed calculation. These are detailed below.

Please note that where equations are given, the notation is similar to that of FORTRAN, with the provision that all variables are of type 'REAL'.

1. Earth's Geometric Shape

The shape of the earth is a "pear" shaped spheroid with a slightly smaller northern hemisphere. The oblate spheroid is usually used as an approximation to the earth's shape. However, for sounding rocket trajectories,

where the range of the rocket is short in comparison to the circumference of the earth, a simplified model may be utilized: the "local spherical earth" in which we approximate the earth's shape by a sphere with a radius equal to the geocentric radius at the launch site. In the program, the local geocentric radius of the launch site (RE) is set to 20899262. feet, a value appropriate for Wallops Island.

2. Earth's Gravitational Field

The gravitational potential of the earth is simplified to the inverse square law without harmonics, i.e.:

$$GG = GM/R**3 \qquad \text{(B-1)}$$

where: GG - is the gravitational potential;

GM - is defined as 1.4076576E16 feet**3/seconds**2;

R - is the distance from the geocenter to the vehicle.

The components of the gravitational potential are:

$$GX = -GG*X$$

$$GY = -GG*Y$$

$$GZ = -GG*Z$$

where X, Y, and Z are the geocentric coordinates of the vehicle.

3. Atmosphere

We have selected an atmospheric model which closely approximates the 1962 U.S. Standard Atmosphere. (Reference 6)

The subroutine ATMSPH is called with altitude as the argument. The atmospheric parameters pressure, density, temperature, viscosity, and speed of sound are returned.

4. Earth's Rotation

During the short duration of Period I, the rotation of the earth may be neglected without appreciable error. In Periods II and III, the effect of

the earth's rotation is accounted for in a transformation from a Launch Inertial Coordinate System to an Earth Fixed Coordinate System.

C. COORDINATE SYSTEMS

In the trajectory calculation, five sets of Cartesian coordinates are used. They are discussed in detail below:

1. Inertial Coordinate System (X, Y, Z) (Inertial at Launch):

The origin of the Inertial System is the earth's center. Once the X and Y axes are determined at launch time, the Inertial coordinates are fixed in earth-centered space, and do not rotate with the earth.

- X - on the earth's equatorial plane, pointing to zero longitude at launch.
- Y - on the earth's equatorial plane, pointing to 90° East longitude at launch.
- Z - perpendicular to the equatorial plane, pointing to the North Pole.

2. Earth-Fixed Coordinate System (X_E, Y_E, Z_E):

The origin of the Earth-Fixed coordinate system is the center of the earth. However, unlike the Inertial system, whose origin is also the earth's center, the X_E and Y_E axes of the Earth-Fixed system are the Greenwich and 90° longitudes respectively and rotate with the earth. At the moment of launch, the Earth-Fixed axes coincide with the Inertial axes.

- X_E - on the earth's equatorial plane, always pointing to the Greenwich longitude.
- Y_E - on the earth's equatorial plane, always pointing to 90° East longitude.
- Z_E - perpendicular to the equatorial plane, pointing to the North Pole.

3. Instantaneous-Topocentric Coordinate System (x, y, z):

The origin of the Instantaneous-Topocentric coordinates is the projection point of the moving rocket on the earth's surface, the point at which the geocentric radius vector to the rocket intersects the earth's surface.

x - on the local horizon plane tangent to the instantaneous projection of the rocket, directed along the local geocentric north.

y - on the local horizon plane tangent to the instantaneous projection of the rocket, directed along the local geocentric east.

z - perpendicular to the instantaneous local tangent plane, directed along the geocentric radius vector, and pointing toward the earth's center to complete the right-hand system.

4. Launch Coordinate System (x_L, y_L, z_L):

The origin of the Launch coordinate system is the launch site. The x_L axis is chosen to point in the direction of launch.

x_L - on the launch-tangent plane, pointing in the direction of launch.

y_L - on the launch-tangent plane, pointing normal to the launch azimuth in the direction which with x_L and z_L forms a right-hand system.

z_L - perpendicular to the launch-tangent plane, positive upward from the earth's center.

5. Body Coordinate System (x_B, y_B, z_B):

The origin of the Body coordinate system is the center of mass of the rocket.

x_B - along the rocket principle axis, positive forward.

y_B - normal to the x_B-z_B plane in the direction which completes the right-hand system.

z_B - perpendicular to the x_B axis and contained in the plane of symmetry of the rocket, positive downward.

D. DYNAMIC EQUATIONS OF THE ROCKET

"HYBRID 3-D" simulates the trajectory of the rocket with two models, one for Period I and another for the rest of the flight. The dynamic equations for eac' model are given here. A FORTRAN-like notation is used (all variables are type "REAL").

1. The Equations for Period I, using two linear and one angular degrees of freedom:

```
AXL   = ((THRUST-DRAG)*CS - FNORM*SN)/MASS
AZL   = ((THRUST-DRAG)*SN + FNORM*CS)/MASS-G0
THDD  = (-LSM*FNORM)/INERT
```

where

AXL	-	acceleration in the XL direction, which is downrange at launch
THRUST	-	the thrust of the rocket motor. Found from table look-up, with time as the argument
DRAG	=	CDS*Q
Q	=	(DENS*VL**2)/2.0
VL	=	SQRT(VXL**2 + VZL**2)
CDS	-	is given by sub-routine TAB1 which enters a look-up table with Mach number as the argument to find CD, the coefficient of drag, and multiplies this by SAREA, the reference area
Q	-	is called the "dynamic pressure".
DENS	-	is given by the Standard Atmosphere sub-routine 'ATMSPH', entered with altitude as the argument. This sub-routine also gives

TEMP	-	temperature
PRES	-	atmospheric pressure
VISC	-	viscosity
SOUND	-	the speed of sound.

VXL, VZL	-	are velocity components in the downrange and vertical directions. Found by integration of AXL and AZL.
CS	=	COS(THETA)
THETA	-	is the inclination angle of the rocket, the angle between the principal body axis and the horizontal. It is originally set to the launch elevation angle, and is modified by double integration of THDD, THETA double dot.
FNORM	=	CNA*SAREA*Q*ALPH
CNA	-	is the slope of the coefficient of the normal force acting on the center of pressure.
SAREA	-	is the reference area.
Q	-	is defined above.
ALPH	-	THETA - GAMA, and is called the "angle of attack". The program switches from the Period I model to the model used for the rest of the flight when ALPH becomes zero (found by interpolation as it crosses from plus to minus). The duration of Period I is about 0.8 second for a NIKE-CAJUN, and about 5 seconds for a SCOUT.
GAMA	=	ATAN (VZL/VXL)
		GAMA is the flight path angle which defines the di[illegible] in which the rocket is moving
MASS	-	is the weight of the rocket divided by the force of gravity.
AZL	-	is acceleration in the Z_L direction, which is vertical at the launch site.
THDD	-	is THETA double dot, the acceleration in the rocket's inclination angle to the X_L axis.
LSM	=	LCP-LCG.
LCP	-	is the distance from the reference position (usually the nose of the rocket) to the center of pressure.
LCG	-	is the distance from the reference position to the rocket's center of gravity.
INERT	-	is the rocket's pitch moment of inertia.

The equations above have been simplified by the elimination of effects such as pitch damping and jet damping which were found to have no significant

effect on the solution, since Period I is of such short duration. Some variables are made constants, with the same justification. Some of these are: GO; CNA; LSM; and INERT.

2. The Equations for Periods II and III:

The angular motion of the rocket is neglected; only the linear motion is considered. The equations of motion are written in the Inertial Coordinate System.

XDDOT = GX + (THX - UX)/MASS

YDDOT = GY + (THY - UY)/MASS

ZDDOT = GZ + (THZ - UZ)/MASS

XDDOT - is X double dot; the acceleration is the direction of the Inertial X-axis (please refer to Section C for a definition of the various coordinate systems used).

GX - is the X component of the acceleration of gravity, as felt at the rocket's center of gravity.

THX - is the rocket's component of thrust in the X-direction. This component of thrust would normally be taken in the direction of rocket motion, except that it may be modified by the thrust control option. The thrust control option may be specified to hold the thrust direction of the rocket constant after a given time of flight or rocket altitude. This option is useful when the rocket attains an altitude where the atmospheric density is so low that the rocket's gyroscopic stability tends to keep it pointing in a constant direction and there is to be an upper-stage firing (Period III).

UX - is the component of drag in the inertial, X direction.

MASS - is the present weight of the rocket, derived from weight tables, divided by the sea-level acceleration due to gravity.

YDDOT, ZDDOT - are the components of acceleration in the inertial Y and Z directions. Their equations are symmetrical with the one for XDDOT.

3. The Equations for the Launch Rail:

The user may specify the type of launcher to be used for the simulation. The simulated rocket will then be accelerated from rest along the launch rail and will have an initial velocity at release. This initial velocity will decrease the duration of Period I. The equation of motion is:

LACC	=	(THRUST - WEIGHT*SEL - DRAG)/MASS
LACC	-	is acceleration in the direction of the launch rail
THRUST	-	is found by table look-up, with time as the argument
WEIGHT	-	is the rocket weight, found by table look-up; time is the argument.
SEL	-	is the sine of the elevation angle of the launcher.
DRAG	-	is the aerodynamic drag of the rocket. This term is quite small, and could be neglected. The friction drag of the launcher is not included, although the equation might be reformulated to add it to the aerodynamic drag. In the practical case of launch elevation angles of perhaps 80°, the normal force on the launch rail (which is multiplied by the coefficient of friction to give the friction drag) is very low, resulting in negligible friction drag.
MASS	=	WEIGHT/GO
GO	-	is the acceleration due to gravity at sea level.

The initial conditions are entered and then this equation is integrated repeatedly to give the velocity and distance traveled on the launch rail. When this distance equals the length of the launcher, the program switches to solution of the equations for Period I.

E. SUMMARY

A computer program, "HYBRID 3-D", has been described which simplifies trajectory simulation.

A comparison of a "6-D" trajectory, a "standard" 3-d trajectory, and "HYBRID 3-D" is shown in Figure 3. "HYBRID 3-D" achieves good accuracy through the special treatment of the untrimmed period near launch.

"HYBRID 3-D" also contains a simulation of the dynamics of the launch rail to enhance accuracy in simulating the trajectory of slow-accelerating rockets.

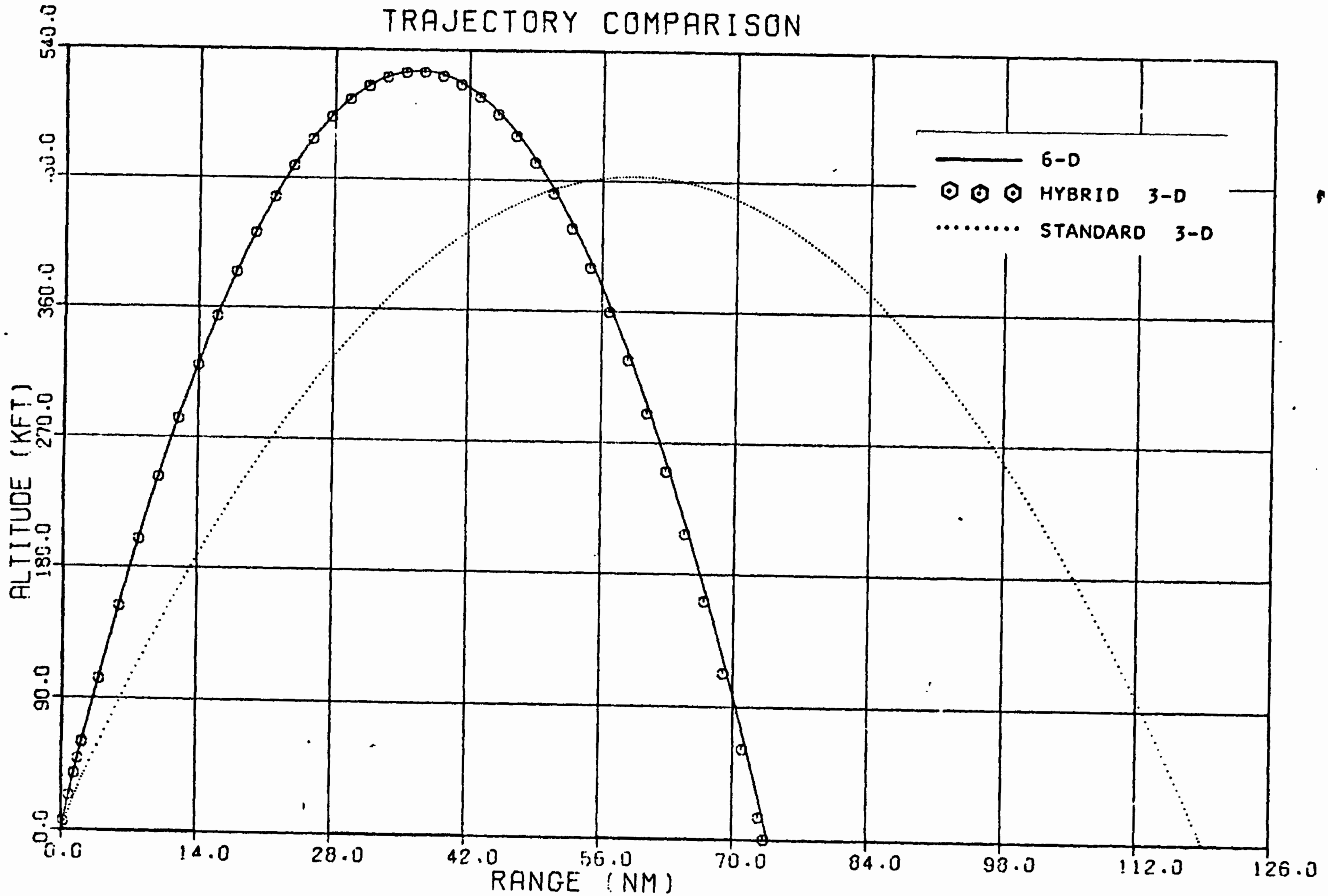

TRAJECTORY COMPARISON
6-D
HYBRID 3-D
STANDARD 3-D
ALTITUDE (KFT)
540.0
360.0
270.0
90.0
0.0
RANGE (NM)
0.0
14.0
28.0
42.0
56.0
70.0
84.0
112.0
126.0

Figure 3.

Part Two

Computer Programming and User's Instruction

TABLE OF CONTENTS

I. INTRODUCTION

This trajectory program is written in Fortran IV and was tested on the Honeywell 625 computer. Input data is transferred from punched cards to disc (file code 05). The program reads this disc, prints the data, rewinds the disc, and rereads the disc as data is needed for calculations. The math model includes aerodynamic drag, rotating earth, launch rail simulation, rigid body dynamics at lift-off, and standard 3D point-mass equations. After leaving the launcher, the rocket is restricted to motion in two dimensions until the angle of attack goes to zero; then, the motion becomes three dimensional with thrust and velocity vectors aligned. The equations of motion are integrated by the Runge-Kutta method, utilizing variable step size for better efficiency. Processing time is approximately 0.7 minutes for each 100 seconds of trajectory time. Output data may be printed in English or metric units, and written on tape (file code 11).

2. PROGRAM VARIABLE DEFINITIONS

A	=	semi-major axis of IIP ellipse
ACX,ACY,ACZ	=	components of acceleration with respect to topocentric system
ACCI	=	inertial acceleration
ALIM	=	altitude where pressure and density are set equal to zero (400,000 ft.)
ALPH	=	angle of attack (α)
ALT	=	altitude
ALTC	=	input altitude at which thrust vector remains in constant direction
AREA	=	rocket nozzle exit area (A_ℓ)
ARRAY	=	matrix for storing all output data (in English units)
ARREY	=	matrix for storing all output data in metric units
AXL	=	acceleration along launch azimuth in launch coordinate system
AZ	=	flight azimuth angle; or launch azimuth (Λ)
AZL	=	acceleration in vertical direction in launch coordinate system
AZLCH,ELLCH	=	look azimuth and look elevation angle with respect to launch site
AZRAD,ELRAD	=	look azimuth and look elevation angle with respect to radar
CAPT	=	time from launch to impact at IIP
CD	=	drag coefficient (C_D)
CDD	=	input table of drag coefficients (C_D); (includes drag area, if SAREA = 1.0)
CNA	=	normal force coefficient
CONV	=	$\pi/180°$
CPHI	=	cos (ϕ)

DELTA	=	range angle (δ)
DENS	=	atmospheric density
DIFF	=	thrust minus drag
DIST	=	distance traveled on launch rail
DRAG	=	drag force (D)
DT	=	integration time interval
EDLCH,NDLCH, ZDLCH	=	east, north, vertical velocity components with respect to launch site
EDRAD,NDRAD, ZDRAD	=	east, north, vertical velocity components with respect to radar
EL	=	flight elevation angle or launch elevation angle (Γ)
ELCH,NLCH, ZLCH	=	east, north, vertical coordinates with respect to launch site
ELPREV	=	previous value of velocity elevation angle
EPBIG	=	maximum relative error allowed in Runge-Kutta method (1.0×10^{-5})
EPS	=	geodetic latitude minus geocentric latitude
EPTINY	=	minimum relative error allowed in Runge-Kutta method (5.0×10^{-7})
ERAD,NRAD, ZRAD	=	east, north, vertical coordinates of rocket with respect to radar
FNORM	=	normal force on rocket
FRAC	=	interpolation fraction for Mach number
GAMA	=	velocity elevation angle (Γ)
GO	=	gravity acceleration at Earth surface (32.174 ft/sec^2)
GDLAT	=	geodetic latitude

GM	=	gravitation constant (1.4076576×10^{16} ft^3/sec^2)
GX, GY, GZ	=	components of gravity acceleration
HIGH	=	maximum integration interval = one half the print time interval
IEND	=	control constant (set = 1 at impact)
IIPLAT,IIPLON IIPR,IIPTIM	=	latitude (GD), longitude, ground range, and impact time for instantaneous impact point
INERT	=	moment of inertia of rocket (I)
INEXT	=	control constant (set = 1 at each phase time)
IOPT	=	thrust reference option (=1 for sea level, =2 for vacuum)
IPRINT	=	control constant (set = 1 when printout is desired)
K	=	matrix for storing K values for Runge-Kutta method
KCON	=	1 for main rocket calculations; 2 for spent stage calculations
KDEL	=	increment for subscript of phase control constant
KPAG	=	output page number
KSPENT	=	present spent stage number
KSTOP	=	0 for no stop at apogee 1 for stopping calculations at apogee
LACC	=	acceleration on launch rail
LAMDA	=	angle from X axis in X, Y plane ($\bar{\lambda}$)
LAMDA0	=	launch longitude (λ_0)
LAMDA1	=	radar longitude (λ_1)
LAT	=	geocentric latitude (ϕ)
LENGTH	=	length of launch rail
LINE	=	number of printout line

LONG	=	longitude (λ)
LOW	=	smallest integration time interval (1.0×10^{-4} sec)
LSM	=	distance from center of gravity to center of pressure (ℓ_{sm})
LVEL	=	velocity on launch rail
MACH	=	Mach number (M)
MASS	=	total rocket mass
MCH	=	input table of Mach numbers corresponding to table of drag coefficients
MESS	=	array for storing printout message
NAP	=	control constant (= 1 at apogee)
NCD	=	number of drag coefficients in table
NDEL	=	increment for subscript of phase time
NGEO	=	0 for geocentric elevation angle and azimuth (input) 1 for geodetic elevation angle and azimuth (input)
NID	=	control constants for phase changes = 1, read phase message only = 2, read phase message and drag table = 3, read phase message and thrust, weight, drag tables = 4, nothing is read in
NIIP	=	input constant for special output tape = 0, tape time interval same as print time interval = 1, tape time interval equal 0.1 sec, and PTI must be an integer multiple of 0.1 sec = 2, no tape output
NMT	=	number of rocket motors
NPAGE	=	1 for output page A 2 for output page B 3 for output page C 4 for output page D 5 for output page E 6 for output page F

NPH =	present phase number
NPHS =	number of phase times
NSEP =	number of motor separations
NSKIP =	input control constant = 0, ALPHA routine performed = 1, ALPHA routine skipped
NSPENT =	number of spent stages left to be calculated
NSTOP =	0 for TSTOP = 0.0 (no termination on time) 1 for TSTOP ≠ 0 (trajectory terminates on time)
NSYS =	output control constant 1, for English units 2, for metric units 3, both English and metric
NTH =	number of thrust values in table
NWT =	number of propellant weights in table
OMEGA =	earth rotation speed (7.29211×10^{-5} rad/sec) (ω)
PO =	atmospheric pressure at surface of earth (2115.666 lbs/ft^2)
PHAS =	output message in column 2 to 7
PHI =	angle from X, Y plane to R vector (ϕ)
PHI0 =	launcher geodetic latitude
PHI1 =	radar site geodetic latitude
PHT =	phase times (input)
PI =	3.14159265 (π)
PI2 =	2π
PRES =	atmospheric pressure (p)
PTI	print time interval
PTIME =	print time

Q =	dynamic pressure
R =	distance from earth center to rocket
RA =	equatorial radius (20925741 ft)
RANGE =	ground range from launch site to rocket
RB =	polar radius (20855591 ft)
RE =	earth radius (20899262 ft)
RLCH =	slant range from launch site to rocket
RRAD =	slant range from radar to rocket
S =	matrix for storing X, Y, Z, $\dot{X}$, $\dot{Y}$, $\dot{Z}$, $\ddot{X}$, $\ddot{Y}$, $\ddot{Z}$ in Runge-Kutta routine
SAREA =	drag area (= 1.0 when area is included in drag coefficient)
SA =	matrix for storing all output data
SMAT =	transformation matrix, topocentric to inertial
SOUND =	speed of sound
SPHI =	sin (ϕ)
SPS =	array for storing initial spent stage variables
STPTI =	stored value of the print time interval
TO =	launch time = 0.0 sec (t_0)
T1 =	time to begin constant thrust direction in inertial system
TEMP =	atmospheric temperature
TFRAC =	time interpolation fraction
THCON =	1 for no thrust control 2 for constant thrust direction beginning at time T1, or at altitude ALTC
THDD =	angular acceleration in launch coordinate system
THETA =	thrust elevation angle
THREF =	thrust at sea level or in vacuum

THRUST =	rocket thrust (TH)
THS =	thrust table input
TIM =	time table corresponding to thrust table
TIM1, TIM2 =	times used to control writing output tape
TIME =	time from launch (t)
TMAT =	transformation matrix, inertial to topocentric
TSEP =	table of separation times (for single stage rocket, set TSEP = 0.0)
TSTOP =	time to stop calculations
TWT =	time table corresponding to propellant weight table
VEL =	total speed of rocket (inertial)
VELLCH =	rocket speed with respect to launch site
VELRAD =	rocket speed with respect to radar
VISC =	atmospheric viscosity
VT =	total velocity (topocentric)
VX, VY, VZ =	topocentric velocity components
WEIGHT =	total rocket weight
WPL =	payload weight
WPP =	present propellant weight
WPP2 =	total propellant weight of all unfired motors
WPR =	input table for propellant weight
WRM =	present rocket motor weight
WTM =	table of rocket motor weights
WTP =	initial propellant weight in each rocket
WTS =	spent stage weight
X, Y, Z =	inertial coordinates

XDOT, YDOT, ZDOT = inertial velocity components ($\dot{X}$, $\dot{Y}$, $\dot{Z}$)

XDDOT, YDDOT, ZDDOT = inertial acceleration components ($\ddot{X}$, $\ddot{Y}$, $\ddot{Z}$)

XIP, YIP, ZIP = inertial position of the instantaneous impact point

XL, YL, ZL = launch system coordinates where XL is along launch azimuth

XT, YT, ZT = topocentric coordinates

3. RUNGE-KUTTA INTEGRATION ROUTINE

For a discussion of the Runge-Kutta formula of order 4, see Reference 7.

Here, only the equations in X are listed, since the Y and Z equations correspond exactly.

(NOTE: the actual subroutine uses matrix S to store all the inertial variables, X, Y, Z, $\dot{X}$, $\dot{Y}$, $\dot{Z}$, $\ddot{X}$, $\ddot{Y}$, $\ddot{Z}$)

Step 1: Input X_0, $\dot{X}_0$, $\ddot{X}_0$; time = T_0

$$K_{11} = (DT)\ \dot{X}_0 \quad K_{41} = (DT)\ \ddot{X}_0$$

Step 2: tire = $T_1 = T_0 + DT/2$

$$X_1 = X_0 + (\frac{DT}{2})\ \dot{X}_0$$

$$\dot{X}_1 = \dot{X}_0 + (\frac{DT}{2})\ \ddot{X}_0$$

$$\ddot{X}_1 = \ddot{X}\ (T_1, X_1, Y_1, Z_1, \dot{X}_1, \dot{Y}_1, \dot{Z}_1)$$

$$K_{12} = (DT)\ \dot{X}_1 \quad K_{42} = (DT)\ \ddot{X}_1$$

Step 3: $\quad \text{time} = T_2 = T_0 + \frac{DT}{2} \qquad \text{(NOTE: } T_2 = T_1\text{)}$

$$X_2 = X_0 + \left(\frac{DT}{2}\right) \dot{X}_1$$

$$\dot{X}_2 = \dot{X}_0 + \left(\frac{DT}{2}\right) \ddot{X}_1$$

$$\ddot{X}_2 = \ddot{X}\,(T_2, X_2, Y_2, Z_2, \dot{X}_2, \dot{Y}_2, \dot{Z}_2)$$

$$K_{13} = (DT)\,\dot{X}_2 \qquad K_{43} = (DT)\,\ddot{X}_2$$

Step 4: $\quad \text{time} = T_3 = T_0 + DT$

$$X_3 = X_0 + (DT)\,\dot{X}_2$$

$$\dot{X}_3 = \dot{X}_0 + (DT)\,\ddot{X}_2$$

$$\ddot{X}_3 = \ddot{X}(T_3, X_3, Y_3, Z_3, \dot{X}_3, \dot{Y}_3, \dot{Z}_3)$$

$$K_{14} = (DT)\,\dot{X}_3 \qquad K_{44} = (DT)\,\ddot{X}_3$$

Solution at time $= T_4 = T_0 + DT$

$$X = X_0 + (1/6)\,(K_{11} + 2K_{12} + 2K_{13} + K_{14})$$

$$\dot{X} = \dot{X}_0 + (1/6)\,(K_{14} + 2K_{42} + 2K_{43} + K_{44})$$

$$\ddot{X} = \ddot{X}(T_4, X, Y, Z, \dot{X}, \dot{Y}, \dot{Z})$$

<u>Error Analysis</u>:

The Runge-Kutta method of order 4 will result in an error of order 5, which is reduced by using "extrapolation to the limit".

First, find a solution using DT = H; then find a corresponding solution by using DT = (H/2) twice. Finally, combine the two solutions in such a way that most of the order 5 error is eliminated.

X_{EX} = exact solution

X(1) = solution using DT = H

X(2) = solution using DT = H/2 twice

$$X(1) = X_{EX} + AH^5 + \text{order } 6$$

$$X(2) = X_{EX} + B(H/2)^5 + C(H/2)^5 + \text{order } 6$$

The factors A, B, and C are composed of derivatives of the acceleration function. The X(2) solution has two order 5 errors, because twice as many steps are required when DT = (H/2).

If H is small, A, B, and C will be approximately equal, resulting in:

$$X(1) - X_{EX} \simeq AH^5$$

$$X(2) - X_{EX} \simeq \frac{AH^5}{16}$$

$$X_{EX} \simeq X(2) + (1/15)\ [X(2) - X(1)]$$

The right hand side is an improved approximation for the exact solution. Similar equations are used for Y, Z, $\dot{X}$, $\dot{Y}$, $\dot{Z}$.

Optimum DT

In the Runge-Kutta formula, the theoretical error term is of order 5, and the smaller the step size, DT, the smaller this error becomes. However, when DT is made smaller, the number of integration steps increases, resulting in more computer round-off error.

To find an "optimum DT", the program proceeds as follows:

1. Find the velocity components $\dot{X}(1)$, $\dot{Y}(1)$, $\dot{Z}(1)$, using DT = H.

2. Find $\dot{X}(2)$, $\dot{Y}(2)$, $\dot{Z}(2)$, using DT = H/2 twice.

3. Calculate the relative errors in velocity:

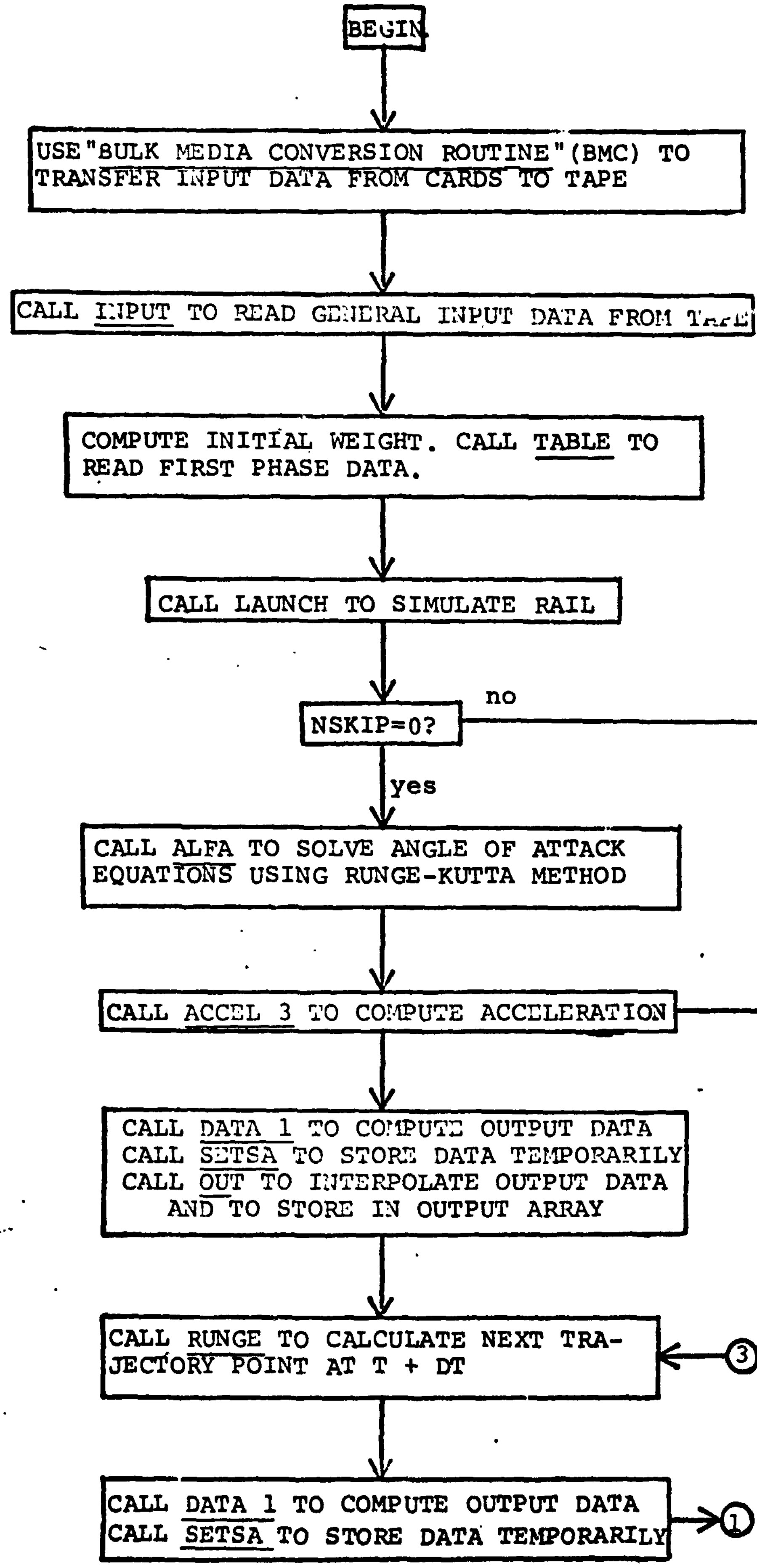
$$E = \frac{|\dot{X}(1) - \dot{X}(2)|}{\dot{X}(2)}, \text{ etc.}$$

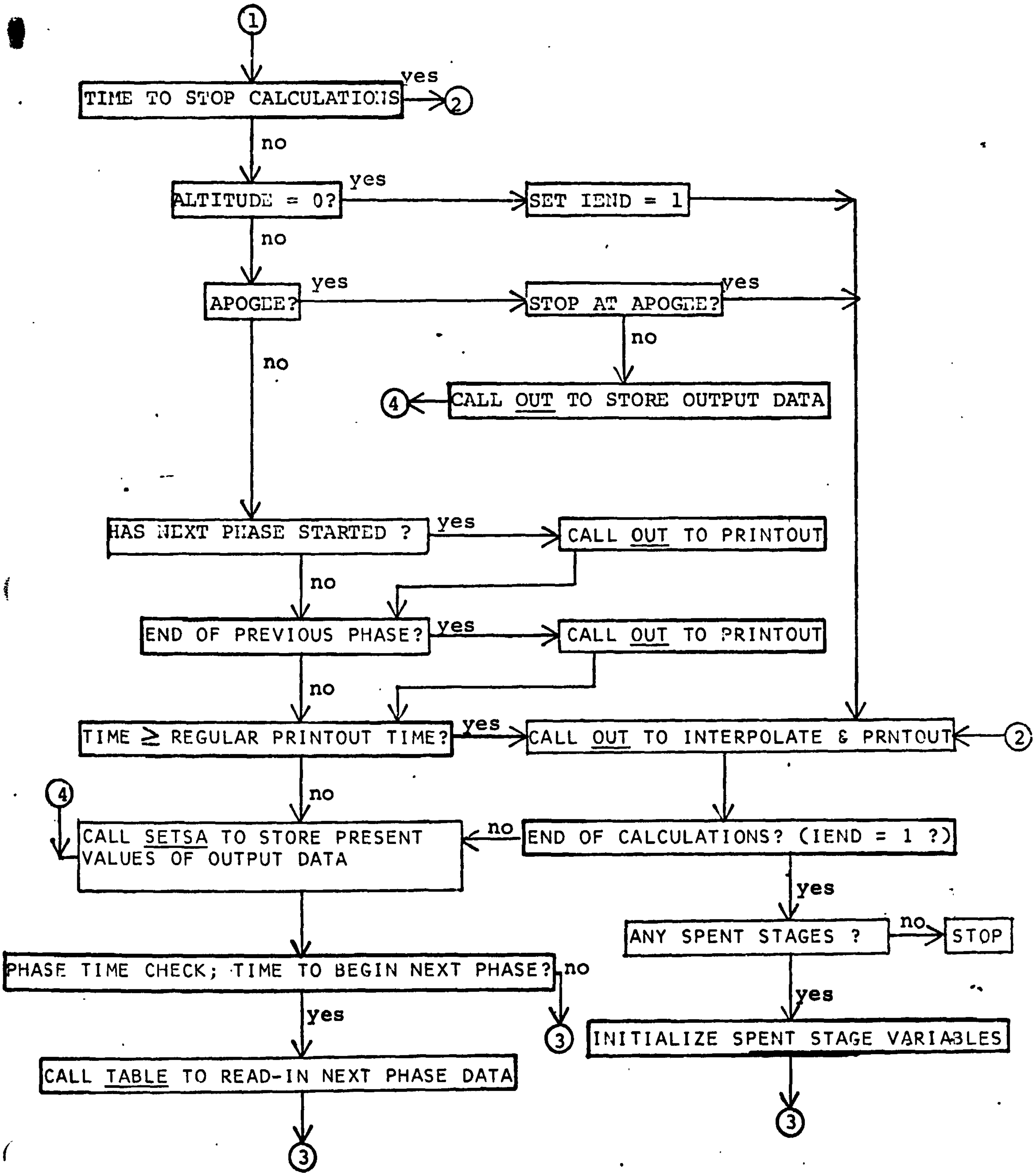
4. If E < EPTINY, then DT is too small, and computer time would be wasted. Thus, DT is increased for the next integration interval.

5. If E > EPBIG, then DT is too large, and the velocity error is unacceptable. Thus, the calculation is repeated using a smaller DT.

   The choice of EPTINY and EPBIG is quite arbitrary. Good results have been obtained with EPTINY = 10^{-7} and EPBIG = 10^{-5}.

4. GENERAL FLOW CHART




1
TIME TO STOP CALCULATIONS
yes
2
no
ALTITUDE = 0?
yes
SET IEND = 1
no
APOGEE?
yes
STOP AT APOGEE?
yes
no
CALL OUT TO STORE OUTPUT DATA
4
no
HAS NEXT PHASE STARTED ?
yes
CALL OUT TO PRINTOUT
no
END OF PREVIOUS PHASE?
yes
CALL OUT TO PRINTOUT
no
TIME ≥ REGULAR PRINTOUT TIME?
yes
CALL OUT TO INTERPOLATE & PRNTOUT
2
no
4
CALL SETSA TO STORE PRESENT VALUES OF OUTPUT DATA
no
END OF CALCULATIONS? (IEND = 1 ?)
yes
ANY SPENT STAGES ?
no
STOP
PHASE TIME CHECK; TIME TO BEGIN NEXT PHASE?
no
3
yes
yes
INITIALIZE SPENT STAGE VARIABLES
CALL TABLE TO READ-IN NEXT PHASE DATA
3
3

5. SUBROUTINE DESCRIPTION

ACCELO --- Subroutine ACCELO computes the relative linear acceleration and angular acceleration, assuming the rocket flies in the launch plane (2 dimensions) with an angle of attack.

ACCEL1 --- Subroutine ACCE11 computes the three components of inertial acceleration which determine the main trajectory.

ACCEL2 --- Subroutine ACCEL2 computes the relative linear acceleration (one dimension) for motion along the launch rail, assuming there is no frictional force.

ACCEL3 --- Subroutine ACCEL3 computes the acceleration using elevation and azimuth rather than velocity components.

ALFA --- Subroutine ALFA determines the trajectory while the rocket flies with an angle of attack during the launch phase. Linear motion is constrained to the launch plane until the angle of attack goes to zero (or until the phase is nearly completed).

ATMSPH --- Subroutine ATMSPH computes temperature, pressure, viscosity, density, and speed of sound using an eighth degree polynomial approximation for the 1962 Standard Atmosphere data.

CNALFA --- Subroutine CNALFA interpolates to find the normal force coefficient and pitch damping coefficient which are used in the angle of attack equations.

DATA1 --- Subroutine DATA1 computes most of the output data.

DIRECT --- Subroutine DIRECT determines the direction of the thrust vector; the thrust is either lined up with velocity, or the thrust maintains a constant direction with respect to the inertial system. Normally, the first option is used; however, for spin stabilized rockets, the second option can be used at high altitudes. (NOTE: This subroutine is not called during the ALFA routine.)

IIP --- Subroutine IIP computes the instantaneous impact points assuming zero thrust and vacuum trajectory.

INPUT --- Subroutine INPUT reads data from input tape 05 and prints it.

LAUNCH --- Subroutine LAUNCH simulates a frictionless launch rail but includes aerodynamic drag in the equations of motion.

MAT --- Subroutine MAT calculates the elements of the transformation matrix for a coordinate system rotation. The inverse matrix is also calculated.

MATLCH --- Subroutine MATLCH calculates the transformation from the inertial system to the launch site system.

MATRAD --- Subroutine MATRAD calculates the transformation from the inertial system to the radar site system.

MATROC --- Subroutine MATROC calculates the transformation from the inertial system to the instantaneous topocentric system.

OUT --- Subroutine OUT interpolates all output data, stores it, and prints it out.

RUNGE --- Subroutine RUNGE integrates the equations of motion, using the fourth order Runge-Kutta method with variable step-size. Refer to Part 2, Section 3.

SETSA --- Subroutine SETSA transfers output data to a temporary storage array.

TABLE --- Subroutine TABLE reads in thrust, weight, and drag tables and phase messages.

TAB1 --- Subroutine TAB1 interpolates to find the drag coefficient.

TAB2 --- Subroutine TAB2 interpolates to find the thrust.

TAB3 --- Subroutine TAB3 interpolates to find the total weight.

TPOUT --- Subroutine TPOUT writes a special output tape with time interval controlled by NIIP. (See input description.)

TRANS1 --- Subroutine TRANS1 transforms from the inertial system to one of the rotating systems.

TRANS2 --- Subroutine TRANS2 transforms from a rotating system to the inertial system.

6. HYBRID 3-D INPUT DESCRIPTION

Input cards must be in FORTRAN NAMELIST format or in the form of a message (see sample); all cards must be punched only in columns 2 through 72. Each input value must be followed by a comma, except the last item in a list, which is followed by a $.

1. The following lists must always be input: NAMQ, NAMR, NAMS, NAMT, NAMU, NAMV. Within a list, if a particular variable is not required, it should be omitted.

2. The lists NAM2, NAM3, NAM4, NAM5 may be input depending upon the trajectory and type of rocket. Thrust, weight, and drag table input is controlled by the phase time array, PHT, and the array NID.

3. All thrust, weight, and drag tables have a limit of 50 values and must have at least two values. The input arrays PHT and NID have a limit of 14 values; the arrays WTM and WTP have a limit of 6 values.

4. For a thrusting phase, the thrust, weight, and drag tables are preceded by one card containing a "phase message"; this may be any phrase to identify the phase; e.g. STAGE 1 THRUSTING. A phase message card must also precede the drag table for a coasting phase. The phase messages may be punched in column 2 through 72, but only the characters in column 2 through 7 will appear on the output (in column 2 through 7) (See sample input and output.)

5. To simulate the rocket launcher, set LENGTH equal to the length of the launch rail. This option restricts the motion to one dimension only.

6. To simulate angle of attack near lift-off, set NSKIP = 0. Motion will be in two dimensions until the angle of attack goes to zero; then, the program switches to three dimensions, with thrust and velocity vectors aligned. (Angle of attack can be calculated only during first stage thrusting.)

7. If NSKIP = 1, then the angle of attack routine is skipped, and the motion will be in three dimensions with thrust and velocity aligned.

8. The trajectory terminates (1) at altitude equal zero; (2) at apogee; (3) or at a given time. See the control constants NSTOP, TSTOP, KSTOP. If the trajectory is not stopped at apogee or at a given time, then it terminates when altitude equals zero.

9. For tape output, see control constant NIIP. **In the deck for execution, include a tape control card with file code 11.**

INPUT DESCRIPTION

FIRST CARD: title phrase in column 2 to 72

NAMQ List:

$NAMQ in column 2 through 6 followed by:

VT	=	initial speed (ft/sec)
EL	=	initial flight elevation angle or launch elevation angle (deg.)
AZ	=	initial flight azimuth angle or launch azimuth angle (deg.)
NGEO	=	0 if EL, AZ are geocentric angles 1 if EL, AZ are geodetic angles
PHI	=	initial geodetic latitude (deg.)
LAMDA	=	initial longitude (deg.)
TIME	=	initial time (not less than 0.0), (sec) (launch time should equal zero)
ALT	=	initial altitude (ft.), (NOTE: launch altitude should equal 0.0 only)
LENGTH	=	length of launch rail (ft.)

NAMR List:

$NAMR in column 2 through 6 followed by:

PTI	=	print time interval (not less than .01), (sec)
NSPENT	=	number of spent stages to be run
NSTOP	=	1 if TSTOP not equal 0.0 (otherwise, omit)
TSTOP	=	time at which calculations are terminated (sec.), (if not desired, omit)
NSKIP	=	0 alpha routine calculations (beginning at launch) 1 alpha routine skipped
KSTOP	=	1 to stop calculations at apogee (otherwise, omit)
THCON	=	1 for no thrust control 2 for constant thrust direction beginning at time = T1 or at altitude ALTC (the constant direction is determined by the program)

T1 = time to begin constant thrust direction (sec.), (if not used, omit)

ALTC = altitude where thrust control begins (ft.), (if not used, omit)

NSYS = 1 for English output
2 for metric output
3 for both English and metric output

NIIP = 0, special tape time interval same as print time interval
1, tape time interval equal 0.1 sec., and PTI equal an integer multiple of 0.1 sec.
2, no tape output

NPAGE = 1 for output page A
2 for output page B
3 for output page C
4 for output page D
5 for output page E
6 for output page F

[e.g., if NPAGE = 2, 4, 5; pages B, D, E will be printed out]

<u>NAMS List</u>:

$NAMS in column 2 through 6 followed by:

PHIO = launcher geodetic latitude (deg.)

LAMDAO = launcher longitude (deg.)

PHI1 = radar site geodetic latitude (deg.) (if not used, omit)

LAMDA1 = radar site longitude (deg.) (if not used, omit)

<u>NAMT List</u>:

$NAMT in column 2 through 6 followed by:

NPHS/PHT= phase time array (sec.) (e.g. 1st stage ignition time, 1st stage burnout time, 2nd stage ignition, 2nd stage burnout, etc.)

NTAB/NID = array of control constants for phase changes

1, to read in phase message only
2, to read phase message and drag table
3, to read in phase message and thrust, weight, and drag tables
4, nothing is read in

[NOTE: There must be a value of NID for each value of PHT, plus one additional value corresponding to apogee.]

NAMU List:

$NAMU in column 2 through 6 followed by:

NMT/WTM = list of inert rocket motor weights (lbs.) (includes miscellaneous weight)

WPL = payload weight (lbs.)

NSEP/TSEP = table of motor separation times (sec.)
[NOTE: For single stage rocket, omit]
[Each value of TSEP must appear in the PHT array.]

WTP = list of propellant weights (one weight for each stage), (lbs.)

NAMV List:

$NAMV in column 2 through 6 followed by:

LSM = distance from center of gravity to center of pressure (ft.)

INERT = moment of inertia (slug $ft.^2$)

CNA = normal force coefficient (rad^{-1})

NAM2 List: (thrust table, input if NID = 3)

$NAM2 in column 2 through 6 followed by:

AREA = rocket nozzle exit area ($ft.^2$)

IOPT = 1 for sea level thrust table,
2 for vacuum thrust table

TIM = time table corresponding to thrust table (sec.)

NTH/THS = thrust table (lbs.)

NAM3 List: (propellant weight table, input if NID = 3)

$NAM3 in column 2 through 6 followed by:

TWT = time table corresponding to weight table (sec.)

NWT/WPR = propellant weight table (lbs.)

NAM4 List: (drag data, input if NID = 2 or 3)

$NAM4 in column 2 through 6 followed by:

SAREA = drag area (ft.^{2})
(=1.0 when the area is included in the drag coefficient)

MCH = table of Mach numbers corresponding to drag coefficient table

NCD/CDD = table of drag coefficients

NAM5 List: (input if NSPENT > 0)

$NAM5 in column 2 through 6 followed by:

PTI = print time interval for spent stage printout (sec.)

WTS = spent stage weight (lbs.)

NTAB/NID = array of phase control constants for spent stage trajectory:

1, to read phase message only
2, to read phase message plus drag table
4, nothing is read in

NOTE: there should be one or two values for NID--one value for the start of the spent stage trajectory and a second value for apogee. If the trajectory begins after apogee, then only one value is needed.

Last card of input data:

Z99999 in column 7 through 12

7. HYBRID 3-D OUTPUT DESCRIPTION

Printed output can be in English and/or metric units by setting the input constant NSYS. In addition, there are six different page options which can be selected using the array NPAGE. Tape output is in English units only. Printout peculiarities are listed below:

1. If the launch rail simulation is used (when LENGTH > 0), output will be suppressed while the rocket moves along the rail.

2. At phase changes, there is double printout plus a phase message from the input data.

3. Built-in messages include:

   TSTOP, printed when the trajectory is terminated at a given time.

   APOGEE, printed out near the actual apogee.

   ALT=0, printed at the end of the trajectory when the altitude equals zero.

4. IIP output is suppressed for spent stage trajectories.

5. Above the atmosphere limit (400,000 ft.), the MACH number will be blank.

Notes on Metric Units:

nt = newton	lbs. = pounds	1 ft.	= 0.3048 m
kg = kilogram	ft. = feet	1 N.M.	= 1.852 km
m = meter	N.M. = nautical mile	1 lb.	= 4.4482 nt

1 slug = 14.594 kg

1 lb./ft^2 = 47.880 nt/m^2

PAGE A:

TIME	=	time from launch (sec.)
ALPHA	=	angle of attack (deg.)
TH EL	=	thrust elevation angle (deg.)
FL EL	=	velocity elevation angle (deg.)
FL AZ	=	velocity azimuth angle (deg.)
ALT	=	altitude (ft.)
RANGE	=	surface range (N.M.)
LAT GD	=	geodetic latitude (deg.)
LONG	=	longitude (deg.)

PAGE B:

TIME	=	time from launch (sec.)
THRUST	=	rocket thrust (lbs.)
WEIGHT	=	rocket weight (lbs.)
DRAG	=	aerodynamic drag (lbs.)
MACH	=	Mach number (suppressed when altitude > 400,000 ft.)
DYN PR	=	dynamic pressure ($lb./ft.^2$)
REL ACC	=	acceleration relative to earth ($ft./sec.^2$)
REL VEL	=	velocity relative to earth (ft./sec.)
MASS	=	mass of rocket (slugs)

PAGE C: Launch Site Coordinate System

TIME	=	time from launch (sec.)
XL,YL,ZL	=	position coordinates (ft.)
VXL,VYL, VZL	=	velocity components (ft./sec.)
RXY	=	tangent plane range (ft.)
GAML	=	velocity elevation angle with respect to tangent plane (deg.)
VEL-L	=	velocity with respect to launch site (ft./sec.)
RLDOT	=	rate of change of slant range

PAGE D: Radar Site Coordinate System

TIME	=	time from launch (sec.)
SL RANGE	=	slant range from radar (N.M.)
LOOK AZ	=	look azimuth (deg.)
LOOK EL	=	look elevation angle (deg.)
VEL	=	velocity relative to radar (ft./sec.)
EAST	=	distance to the East (+) or West (-) in the tangent plane (N.M.)
NORTH	=	distance to the North (+) or South (-) in the tangent plane (N.M.)
VERT	=	distance (+) above the tangent plane (N.M.)
VEL-E	=	East (+) or West (-) component of velocity (ft./sec.)
VEL-N	=	North (+) or South (-) component of velocity (ft./sec.)
VEL-V	=	vertical component of velocity (ft./sec.)

PAGE E: Instantaneous Impact Points (Vacuum Trajectory)
[NOTE: Page E is suppressed for spent stages]

TIME	=	time from launch (sec.)
LAT GD	=	geodetic latitude of IIP (deg.)
LONG	=	longitude of IIP (deg.)
RANGE	=	range of IIP from launch site (N.M.)
IP TIME	=	time from launch to instantaneous impact (sec.)

PAGE F: Inertial Coordinate System

TIME	=	time from launch (sec.)
R	=	distance from earth center (ft.)
VEL	=	inertial velocity (ft./sec.)
ACCEL	=	inertial acceleration (ft./sec.^{2})

SPECIAL TAPE OUTPUT

This tape is high density (800 BPI) and contains 71 words per record, as listed below (those not listed are equal to zero):

Word Number	Contents (English Units)
1	TIME
2	WEIGHT
3	THRUST
4	VEL (inertial)
5	R (inertial)
6	ALT
7	RANGE
8	MACH
9	Q
10	LAT GD
12	LONG
14	ALPHA
18	FL EL
20	FL AZ
23	TH EL
37	REL VEL
55, 56, 57	XL, YL, ZL
58, 59, 60	VXL, VYL, VZL
61	RXY
62	GAML
63	VEL-L
64	RLDOT
65	IP TIME (IIP)
66	LAT GD (IIP)
67	LONG (IIP)
68	RANGE (IIP)

8. Deck Set-Up with Object Program on Tape

```
$    IDENT     153200, HYBRID 3D
$    EXECUTE
$    LIMITS    100, 30K, 0, 10K
$    TAPE      R*, X1D,, 3969
$    DATA      05
$    INCODE    IBMF

(data cards)

$    ENDJOB
***EOF
```

Deck Set-Up with Object Program on Cards

```
$    IDENT     153200, HYBRID 3D
$    OPTION    FORTRAN

(object deck)

$    EXECUTE
$    LIMITS    100, 30K, 0, 10K
$    DATA      05
$    INCODE    IBMF

(data cards)

$    ENDJOB
***EOF
```

9. SAMPLE INPUT AND OUTPUT

This is a Nike-Cajun trajectory simulation using a 2-inch launch rail and initial velocity equal to zero. Four phases are shown: stage 1 thrusting, stage 2 coasting, stage 2 thrusting, and stage 2 coasting. Input tables must be in the order specified by the NTAB/NID array:

3 for stage 1 thrust; weight, drag input (NAM2, NAM3, NAM4)
2 for stage 2 drag input for coasting (NAM4)
3 for stage 2 thrust; weight, drag input (NAM2, NAM3, NAM4)
2 for stage 2 drag input for coasting (NAM4)
4 for no input at apogee

The beginning of each phase is given by the NPHS/PHT array:

-0.017 sec for stage 1 ignition
3.523 for stage 1 burnout (and separation)
17.0 for stage 2 ignition
22.0 for stage 2 burnout

Note that the times used for the thrust and weight tables are <u>relative</u> times which are added to the phase times by the program. If the first stage thrust table begins with a value less than the lift-off weight, then the ignition time must be adjusted so that lift-off will occur at zero time.

```
        NIKE-CAJUN TRAJECTORY SIMULATION
$NAMQ   VT = 0.0,      EL = 80.0,  AZ = 120.0, NGFO = 1,
        PHI = 37.8480,  LAMDA = -75.4736, TIME = 0.0, ALT = 0.0,
        LENGTH = .1667$
$NAMR   PTI =0.5,  NSPENT = 0, NSTOP=1,  TSTOP =  25.0,  NSKIP=0,
        KSTOP = 0,  THCON = 1,  TI=0.0,  ALTC=0.0,
        NSYS=1,  NIIP=0,  NPAGE= 1,2$
$NAMS   PHIO = +37.8480,  LAMDAO = -75.4736,
        PHI1 = +37.8412,  LAMDA1 = -75.4855$
$NAMT   NPHS/PHT =  -.017,  3.523,  17.0,  22.0,
        NTAB/NID =      3,      2,       3,      2,     4$
$NAMU   NMT/WTM =  564.0, 97.4,     WPL = 50.0,
        NSEP/TSEP = 3.523,
        WTP = 738.0, 119.0$
$NAMV   LSM = 2.38,  INERT = 1500.0,  CNA = 15.47$
1ST TH         STAGE 1 THRUSTING
$NAM2   AREA = 1.5,  IOPT = 1,
        TIM =      0.0,     0.01,      0.04,      0.05,      0.09,
                   .15,     0.84,      1.14,      1.74,      2.04,
                  2.34,     2.49,      2.64,      2.79,      2.99,
                  3.09,     3.21,      3.28,      3.34,      3.40,
                  3.46,     3.54,
        NTH/THS =    1923.,   26439.,   39417.,   41628.,   42397.,
                   42589.,   43263.,   43551.,   44705.,   45667.,
                   46820.,   46723.,   45570.,   42974.,   37975.,
                   33648.,   24034.,   16343.,   10864.,    6538.,
                    3268.,       0.$
$NAM3   TWT =      0.0,     0.01,      0.04,      0.05,      0.09,
                   .15,     0.84,      1.14,      1.74,      2.04,
                  2.34,     2.49,      2.64,      2.79,      2.99,
                  3.09,     3.21,      3.28,      3.34,      3.40,
                  3.46,     3.54,
        NWT/WPR =  738.0,  737.26,  732.10,   729.98,   721.20,
                  707.87, 553.07,  495.01,   346.63,   275.78,
                  203.27, 166.60,  130.42,    95.72,    53.41,
                   34.69,  16.60,    9.22,     4.95,     2.22,
                     .68,    .00$
$NAM4   SAREA = 1.4741,
           MCH =   0.,     0.75,    1.00,    1.20,    1.60,
                   2.0,    2.40,    2.80,    3.30,    4.00,
                   5.0,
```

```
          NCD/CDD =  .675,   .595,   .925,   .870,   .780,
                     .710,   .660,   .615,   .565,   .520,
                     .455$
COAST          STAGE 2 COASTING
$NAM4     SAREA = .23,
               MCH = 1.0,    2.0,    3.0,    4.0,    5.0,
                     6.0,    8.0,
          NCD/CDD = .93,     .72,    .59,    .51,    .45,
                    .42,     .36$
2ND TH         STAGE 2 THRUSTING
$NAM2     AREA = .223,  IOPT = 1,
          TIM   =   0.0,     0.04,   0.08,   0.15,   0.50,
                    .70,     1.00,   1.20,   1.40,   1.60,
                    2.00,    2.40,   2.80,   2.98,   3.04,
                    3.16,    3.26,   3.40,   3.50,   5.00,
          NTH/THS = 0.,      2050.,  7400.,  7200.,  7075.,
                    7200.,   7700.,  8050.,  8275.,  8400.,
                    8500.,   8725.,  9370.,  9050.,  6900.,
                    2700.,   900.,   225.,   50.,    0.0$
$NAM3     TWT   =   0.0,     0.04,   0.08,   0.15,   0.50,
                    .70,     1.00,   1.20,   1.40,   1.60,
                    2.00,    2.40,   2.80,   2.98,   3.04,
                    3.16,    3.26,   3.40,   3.50,   5.00,
          NWT/WPP = 119.0,   118.80, 117.91, 115.50, 103.73,
                    97.0,    86.46,  79.03,  71.34,  63.48,
                    47.54,   31.30,  14.24,  6.43,   4.17,
                    1.46,    0.61,   0.24,   0.17,   0.0$
$NAM4     SAREA = .23,
               MCH = 1.0,    2.0,    3.0,    4.0,    5.0,
                     6.0,    8.0,
          NCD/CDD = .78,     .62,    .53,    .46,    .42,
                    .38,     .34$
COAST          STAGE 2 COASTING
$NAM4     SAREA = .23,
               MCH = 1.0,    2.0,    3.0,    4.0,    5.0,
                     6.0,    8.0,
          NCD/CDD = .93,     .72,    .59,    .51,    .45,
                    .42,     .36$
        Z99999
     ***EOF
```

NIKE-CAJUN TRAJECTORY SIMULATION

EL = 80.000 AZ =120.000 PAYLOAD = 50.0

	TIME (SEC)	ALPHA (DEG)	TH EL (DEG)	FL EL (DEG)	FL AZ (DEG)	ALT (FT)	RANGE (N.M.)	LAT GD (DEG)	LONG (DEG)
1ST TH	0.50	0.295	80.038	79.743	119.076	104.	0.00	37.8480	-75.4735
	1.00	0.	79.624	79.624	119.081	432.	0.01	37.8479	-75.4734
	1.50	0.	79.480	79.480	119.099	1002.	0.03	37.8478	-75.4730
	2.00	0.	79.382	79.382	119.115	1832.	0.06	37.8475	-75.4726
	2.50	0.	79.309	79.309	119.129	2945.	0.09	37.8473	-75.4719
	3.00	0.	79.253	79.253	119.143	4358.	0.13	37.8469	-75.4711
	3.50	0.	79.204	79.204	119.157	5978.	0.18	37.8465	-75.4702
	3.52	0.	79.202	79.202	119.158	6053.	0.19	37.8465	-75.4702
COAST	3.52	0.	79.202	79.202	119.158	6053.	0.19	37.8465	-75.4702
	4.00	0.	79.155	79.155	119.171	7578.	0.24	37.8461	-75.4693
	4.50	0.	79.103	79.103	119.185	9130.	0.28	37.8457	-75.4684
	5.00	0.	79.050	79.050	119.198	10638.	0.33	37.8453	-75.4675
	5.50	0.	78.995	78.995	119.212	12105.	0.38	37.8449	-75.4666
	6.00	0.	78.938	78.938	119.226	13534.	0.42	37.8445	-75.4658
	6.50	0.	78.879	78.879	119.240	14926.	0.47	37.8442	-75.4650
	7.00	0.	78.818	78.818	119.254	16286.	0.51	37.8438	-75.4642
	7.50	0.	78.755	78.755	119.267	17614.	0.56	37.8435	-75.4634
	8.00	0.	78.690	78.690	119.281	18912.	0.60	37.8431	-75.4626
	8.50	0.	78.624	78.624	119.295	20183.	0.64	37.8428	-75.4618
	9.00	0.	78.556	78.556	119.308	21427.	0.68	37.8424	-75.4611
	9.50	0.	78.485	78.485	119.322	22646.	0.72	37.8421	-75.4603
	10.00	0.	78.413	78.413	119.336	23842.	0.76	37.8418	-75.4596
	10.50	0.	78.339	78.339	119.349	25014.	0.80	37.8415	-75.4588
	11.00	0.	78.263	78.263	119.363	26165.	0.84	37.8411	-75.4581
	11.50	0.	78.185	78.185	119.377	27296.	0.88	37.8408	-75.4574
	12.00	0.	78.105	78.105	119.390	28407.	0.92	37.8405	-75.4567
	12.50	0.	78.023	78.023	119.404	29498.	0.96	37.8402	-75.4560
	13.00	0.	77.939	77.939	119.417	30572.	0.99	37.8399	-75.4553
	13.50	0.	77.853	77.853	119.431	31628.	1.03	37.8396	-75.4547
	14.00	0.	77.765	77.765	119.444	32667.	1.07	37.8393	-75.4540

	TIME (SEC)	THRUST (LBS)	WEIGHT (LBS)	DRAG (LBS)	MACH	DYN PR (LBS/FT2)	REL ACC (FT/SEC2)	REL VEL (FT/SEC)	MASS (SLUG)
1ST TM	0.50	42959.	1456.	212.	0.39	226.95	912.97	433.33	45.3
	1.00	43483.	1343.	1006.	0.82	987.45	985.76	908.61	41.8
	1.50	44392.	1228.	2956.	1.28	2356.86	1053.62	1415.30	38.2
	2.00	45801.	1112.	4874.	1.79	4426.06	1153.00	1966.27	34.6
	2.50	46844.	990.	7192.	2.35	7319.83	1256.47	2573.30	30.8
	3.00	37283.	879.	9409.	2.59	10538.62	988.97	3157.46	27.3
	3.50	1577.	831.	9590.	3.05	11016.34	341.64	3312.62	25.8
	3.52	0.	830.	9521.	3.04	10935.32	400.50	3304.37	25.8
COAST	3.52	0.	266.	1475.	3.04	10934.85	209.80	3304.33	8.3
	4.00	0.	266.	1341.	2.97	9815.74	193.60	3208.34	8.3
	4.50	0.	265.	1221.	2.90	8800.05	179.13	3115.27	8.3
	5.00	0.	266.	1119.	2.83	7918.72	166.26	3029.03	8.3
	5.50	0.	265.	1020.	2.77	7149.59	154.79	2948.88	8.3
	6.00	0.	266.	939.	2.71	6475.07	144.52	2874.16	8.3
	6.50	0.	266.	858.	2.66	5880.82	135.31	2804.31	8.3
	7.00	0.	265.	790.	2.61	5354.97	127.02	2738.83	8.3
	7.50	0.	266.	728.	2.56	4887.71	119.54	2677.30	8.3
	8.00	0.	266.	672.	2.51	4471.07	112.77	2619.33	8.3
	8.50	0.	266.	621.	2.47	4098.27	106.64	2564.58	8.3
	9.00	0.	266.	575.	2.43	3763.62	101.07	2512.77	8.3
	9.50	0.	266.	533.	2.39	3462.28	95.99	2463.62	8.3
	10.00	0.	265.	494.	2.35	3190.23	91.36	2416.90	8.3
	10.50	0.	266.	459.	2.32	2943.96	87.13	2372.41	8.3
	11.00	0.	266.	427.	2.29	2720.51	83.25	2329.94	8.3
	11.50	0.	265.	398.	2.25	2517.28	79.69	2289.35	8.3
	12.00	0.	266.	371.	2.22	2332.07	76.42	2250.46	8.3
	12.50	0.	265.	346.	2.19	2162.96	73.41	2213.15	8.3
	13.00	0.	266.	323.	2.16	2008.24	70.64	2177.30	8.3
	13.50	0.	265.	302.	2.13	1866.45	68.08	2142.78	8.3
	14.00	0.	266.	292.	2.11	1736.30	65.71	2109.50	8.3

	TIME (SEC)	ALPHA (DEG)	TM EL (DEG)	FL EL (DEG)	FL AZ (DEG)	ALT (FT)	RANGE (N.M.)	LAT GD (DEG)	LONG (DEG)
	14.50	0.	77.675	77.675	119.458	33689.	1.11	37.8390	-75.4533
	15.00	0.	77.583	77.583	119.471	34696.	1.14	37.8387	-75.4526
	15.50	0.	77.488	77.488	119.485	35688.	1.18	37.8384	-75.4520
	16.00	0.	77.391	77.391	119.498	36665.	1.21	37.8381	-75.4513
	16.50	0.	77.293	77.293	119.511	37627.	1.25	37.8378	-75.4507
	17.00	0.	77.191	77.191	119.525	38576.	1.28	37.8375	-75.4500
2ND TM	17.00	0.	77.191	77.191	119.525	38576.	1.28	37.8375	-75.4500
	17.50	0.	77.098	77.098	119.538	39605.	1.32	37.8372	-75.4493
	18.00	0.	77.021	77.021	119.550	40854.	1.37	37.8368	-75.4485
	18.50	0.	76.958	76.958	119.562	42358.	1.43	37.8363	-75.4474
	19.00	0.	76.905	76.905	119.574	44159.	1.50	37.8358	-75.4462
	19.50	0.	76.862	76.862	119.586	46295.	1.58	37.8351	-75.4447
	20.00	0.	76.826	76.826	119.598	48822.	1.67	37.8343	-75.4429
	20.50	0.	76.795	76.795	119.610	51651.	1.78	37.8334	-75.4409
	21.00	0.	76.764	76.764	119.621	54495.	1.89	37.8325	-75.4389
	21.50	0.	76.732	76.732	119.633	57332.	2.00	37.8316	-75.4369
	22.00	0.	76.701	76.701	119.645	60164.	2.11	37.8307	-75.4349
COAST	22.00	0.	76.701	76.701	119.645	60165.	2.11	37.8307	-75.4349
	22.50	0.	76.569	76.569	119.656	62980.	2.22	37.8298	-75.4329
	23.00	0.	76.636	76.636	119.668	65771.	2.33	37.8289	-75.4309
	23.50	0.	76.604	76.604	119.680	68540.	2.44	37.8280	-75.4289
	24.00	0.	76.571	76.571	119.691	71287.	2.55	37.8271	-75.4270
	24.50	0.	76.537	76.537	119.703	74016.	2.65	37.8262	-75.4250
TSTOP	25.00	0.	76.504	76.504	119.715	76727.	2.76	37.8253	-75.4231

	TIME (SEC)	THRUST (LBS)	WEIGHT (LBS)	DRAG (LBS)	MACH	DYN PR (LBS/FT2)	REL ACC (FT/SEC2)	REL VEL (FT/SEC)	MASS (SLUG)
	14.50	0.	266.	264.	2.08	1616.65	63.53	2077.37	8.3
	15.00	0.	265.	247.	2.05	1506.40	61.50	2046.30	8.3
	15.50	0.	265.	231.	2.03	1404.94	59.62	2016.21	8.3
	16.00	0.	266.	217.	2.00	1311.22	57.88	1987.04	8.3
	16.50	0.	265.	204.	1.98	1224.60	56.31	1958.71	8.3
	17.00	0.	265.	192.	1.96	1144.41	54.86	1931.14	8.3
2ND TH	17.00	387.	266.	165.	1.96	1144.39	6.24	1931.13	8.3
	17.50	7456.	251.	216.	2.37	1596.13	896.44	2330.27	7.8
	18.00	8086.	234.	279.	2.86	2201.81	1043.48	2810.08	7.3
	18.50	8729.	215.	342.	3.43	2987.01	1225.00	3379.77	6.7
	19.00	9898.	195.	411.	4.13	3925.55	1369.68	4028.39	6.1
	19.50	9292.	174.	497.	4.99	4996.49	1592.74	4762.78	5.4
	20.00	8746.	153.	557.	5.81	6249.18	1690.00	5635.11	4.8
	20.50	471.	145.	517.	6.06	5928.08	41.72	5852.34	4.6
	21.00	460.	145.	459.	6.06	5167.76	29.74	5834.89	4.6
	21.50	449.	147.	393.	6.06	4506.01	19.93	5823.05	4.6
	22.00	0.	147.	342.	6.06	3929.10	106.05	5815.97	4.6
COAST	22.00	0.	147.	378.	6.06	3928.88	113.80	5815.95	4.6
	22.50	0.	147.	325.	6.01	3368.17	102.30	5762.13	4.6
	23.00	0.	147.	280.	5.96	2891.77	92.49	5713.62	4.6
	23.50	0.	147.	242.	5.91	2486.37	84.11	5669.66	4.6
	24.00	0.	147.	209.	5.86	2140.73	76.94	5629.59	4.6
	24.50	0.	147.	181.	5.82	1845.64	70.81	5592.86	4.6
TSTOP	25.00	0.	147.	156.	5.77	1593.36	65.55	5558.98	4.6

```
      3MAIN          MAIN
      DOUBLE PRECISION TMAT,SMAT
      DOUBLE PRECISION ST
      COMMON/STOR1/ALT,TEMP,PRES,DENS,VISC,SOUND
      COMMON/STOR2/X,Y,Z,R
      COMMON/STOR3/TMAT,SMAT,SPHI,CPHI,CL,SL
      COMMON/STOR4/SAREA,MCH,CDD,CD,MACH,NCD,CDS,DRAG
      COMMON/STOR5/TWT,NWT,WPR,WEIGHT,MASS
      COMMON/STOR6/PO,WPM,WPL,WPP2,WTM
      COMMON/STOR7/XDOT,YDOT,ZDOT,XDDOT,YDDOT,ZDDOT
      COMMON/STOR8/OMEGA,RE,RA,RB,GO,GM,ALIM
      COMMON/STOR9/S,NK
      COMMON/STOR10/ST
      COMMON/STOR11/THCON,TI,ALTC,THX,THY,THZ
      COMMON/STOR12/YL,VYL,PXY,GAML,PLDOT
      COMMON/STOR13/XL,ZL,THETA,VXL,VZL,ALPH,GAMA,CAZ,SAZ
      COMMON/STOR14/LSM,INERT,CNA
      COMMON/STOR15/Q,VX,VY,VZ,VT,EL,AZ,RANGE,LAT,LONG,ACC,GDLAT
      COMMON/STOR16/SA,ARRAY,NN
      COMMON/STOR17/LENGTH,SEL,DIST,LVEL,LACC
      COMMON/STOR18/TFRAC,LINE,PTIME,CONV,IEND,IPRINT,NPAGE,KPAG
      COMMON/STOR19/TIME,TO
      COMMON/STOR20/AREA,IOPT,TIM,NTH,THS,THRUST
      COMMON/STOR21/MESS,PHAS
      COMMON/STOR23/SPHI0,CPHI0,LAMDA0,X0,Y0,Z0
      COMMON/STOR24/PHT,NID,NTAB,NPH,KPH
      COMMON/STOR25/LOW,EPTINY,EPBIG,PTI,PI,PI2,DT,HIGH
      COMMON/STOR26/ KCON,KSPENT
      COMMON/STOR27/IIPLAT,IIPLON,IIPR,IIPTIM
      COMMON/STOR28/RLCH,AZLCH,ELLCH,VELLCH,ELCH,NLCH,ZLCH,EDLCH,
     1 NDLCH,ZDLCH
      COMMON/STOR29/RRAD,AZRAD,ELRAD,VELRAD,ERAD,NRAD,ZRAD,EDRAD,
     1 NDRAD,ZDRAD
      COMMON/STOR30/DELTA,VEL,ACCI
      COMMON/STOR31/SPHI1,CPHI1,LAMDA1
      COMMON/STOR32/NSYS,CON
      COMMON/STOR33/NIIP,NCYC,NLIN
      COMMON/STOR34/KLIN
      COMMON/STOR35/NTAP
      COMMON/STOR36/PRES0,DENS0,TEMP0,SOUND0,K1,CON1,KPREV
      COMMON/STOR37/TSLOPE,HALTB,TEMPB,PRESB
   10 FORMAT(    20X,5H EL =,F7.3,5X,5H AZ =,F7.3,5X,10H PAYLOAD =,
     1 F7.1//)
   11 FORMAT(12A6)
   12 FORMAT(1H1)
```

ORIGINAL PAGE IS POOR

```
   13 FORMAT(5X,12A6//)
      DIMENSION TITLE(12)
      DIMENSION TSLOPE( 9),HALTB( 9),TEMPB( 9),PRESB( 9)
      DIMENSION SMAT(3,3),TMAT(3,3),S(9,6),ST(20,10)
      DIMENSION SPS(15,5)
      DIMENSION SA(10,2,10),ARRAY(10,30,10)
      DIMENSION PHT(14),NID(14),WTM(6),TSEP(6),WTP(6)
      DIMENSION TIM(50),THS(50),TWT(50),WPR(50),MCH(50),CDD(50)
      DIMENSION MESS(30)
      DIMENSION NPAGE(10)
      DIMENSION CON(10,7)
      INTEGER THCON
      REAL IIPLAT,IIPLON,IIPR,IIPTIM
      REAL LAMDA,MASS,  LOW,LONG, LAT, MACH,MCH,LAMDA0,LAMDA1,LBAR
      REAL K1
      REAL LENGTH
      REAL LSM, INERT
      REAL NRAD,NLCH,NDRAD,NDLCH
      DOUBLE PRECISION CCS
      EXTERNAL ACCEL1,ACCEL0,ACCEL3
      NAMELIST /NAMO/VT,FL,AZ,NGEO,PHI,LAMDA,TIME,ALT,LENGTH
      NAMELIST /NAMR/PTI,NSPENT,NSTOP,TSTOP,NSKIP,KSTOP,
     1 THCON,T1,ALTC,NSYS,NIIP,NPAGE
      NAMELIST /NAMS/PHI0,LAMDA0,    PHI1,LAMDA1
      NAMELIST /NAMT/NPHS,PHT,NTAB,NID
      NAMELIST /NAMU/NMT,WTM,WPL,NSEP,TSEP,WTP
      NAMELIST/NAMV/LSM,INERT,CNA
      NAMELIST /NAM5/PTI,WTS,NTAB,NID
      DATA BLKK/020202020202/
      DATA APO/6HAPOGEE/,TST/6HTSTOP /,ALTO/6HALT=0 /
C
C
C        INITIALIZE CONSTANTS
      PI = 3.14159265
      OMEGA = 7.29211E-5
      CONV = PI/180.
      PI2 = 2.*PI
      TO =  .0
      PO = 2115.666
      EPTINY = 5.0E-7
      EPBIG = 1.0E-5
      LOW = 1.0E-4
      GM = 1.4076576E16
      RE = 20899262.
```

```
      RA = 20925741.
      RB = 20855591.
      G0 = 32.174
      ALIM = 2.0E+5
      LINE = 0
      IEND = 0
      NCYC = 0
      NLIN = 0
      KPAG = 0
      KCON = 1
      KSPENT = 0
      NAP =
      LSPT = 0
      DO 141 J=1,14
      PHT(J) = 900000.
  141 CONTINUE
      DO 142 J=1,10
      NPAGE(J) = 8
  142 CONTINUE
      PHAS = BLKK
      DO 14  I=1,10
      DO 14  J=1,7
      CON(I,J) = 1.0
  140 CONTINUE
      CON(5,1) = .3048
      CON(6,1) =  1.852
      CON(1,2) =  4.4482
      CON(2,2) =  4.4482
      CON(3,2) =  4.4482
      CON(5,2) =  47.88
      CON(6,2) =  .3048
      CON(7,2) =  .3048
      CON(8,2) = 14.594
      CON(1,3) = .3048
      CON(2,3) = .3048
      CON(3,3) = .3048
      CON(4,3) = .3048
      CON(5,3) = .3048
      CON(6,3) = .3048
      CON(7,3) = .3048
      CON(9,3) = .3048
      CON(1 ,3) = .3048
      CON(1,4) = 1.852
      CON(4,4) = .3048
```

```
      CON(5,4) = 1.852
      CON(6,4) = 1.852
      CON(7,4) = 1.852
      CON(8,4) = .3048
      CON(9,4) = .3048
      CON(1 ,4) = .3048
      CON(3,5) = 1.852
      CON(1,6) = .3048
      CON(3,6) = .3048
      CON(4,6) = .3048
C
C          READ GENERAL INPUT DATA FROM TAPE
      CALL INPUT
      PRINT 12
      READ(5,11) TITLE
      READ(5,NAMQ)
      READ(5,NAMR)
      READ(5,NAMS)
      READ(5,NAMT)
      READ(5,NAMU)
      READ(5,NAMV)
      PRINT 13, TITLE
      PRINT 10,EL,AZ,WPL
      STPTI = PTI
      IF(NIIP .EQ. 2) NTAP=2
      IF(NIIP .EQ. 0 .OR. NIIP .EQ. 2) GO TO 200
      NCYC = 10.*PTI
      PTI =   .1
  200 CONTINUE
      HIGH = PTI/2.
      DTO = PTI/8.
      DTO = AMIN1(0.1,DTO)
      DT = DTO
      EL = EL*CONV
      AZ = AZ*CONV
      PHIO = PHIO*CONV
      EPS = PHIO
      PHI1 = PHI1*CONV
      PHI = PHI*CONV
      LAMDA  = LAMDA0*CONV
      LAMDA1 = LAMDA1*CONV
      LAMDA = LAMDA*CONV
      CAZ = COS(AZ)
      SAZ = SIN(AZ)
```

```
C
C         GEODETIC TO GEOCENTRIC CONVERSIONS
      ARG =(RB/RA)**2*SIN(PHI0)/COS(PHI0)
      PHI0 = ATAN(ARG)
      EPS = EPS - PHI0
      ARG = (RB/RA)**2*SIN(PHI)/COS(PHI)
      PHI = ATAN(ARG)
      ARG = (RB/RA)**2*SIN(PHI1)/COS(PHI1)
      PHI1 = ATAN(ARG)
      SEL = SIN(EL)
      SELGD = SEL
      CEL = COS(EL)
      IF(NGEO .EQ. 0) GO TO 250
      CEPS = COS(EPS)
      SEPS = SIN(EPS)
      SEL = SEL*CEPS - SEPS*CEL*CAZ
      EL = ATAN2(SEL,SQRT(1.0 - SEL**2))
      AZ = ATAN2(CEL*SAZ,CEPS*CEL*CAZ + SEPS*SELGD)
      IF(AZ .LT. 0.0) AZ = AZ + PI2
      CAZ = COS(AZ)
      SAZ = SIN(AZ)
  250 CONTINUE
C
C         INITIALIZE ROCKET MOTOR WEIGHT
      IF(NMT .EQ. 1) GO TO 316
      SUM =  .0
      DO 31   J = 1,NMT
      SUM = SUM + WTM(J)
  310 CONTINUE
      WRM = SUM
      NMOT = 1
C
C         INITIALIZE WEIGHT OF PROPELLANT
      SUM =  .0
      DO  315 J=1,NMT
      SUM = SUM + WTP(J)
  315 CONTINUE
      WPP2 = SUM
      GO TO 317
  316 WPP2 = WTP(1)
      WRM = WTM(1)
      NMOT = 1
  317 CONTINUE
      KDEL = 1
```

```
      NDEL = 1
      KPH = 1
      NPH = 1
C
C         READ FIRST PHASE DATA
      CALL TABLE
      NPH = NPH + NDEL
      KPH = KPH + KDEL
      SPHI0 = SIN(PHI0)
      CPHI0 = COS(PHI0)
      SPHI1 = SIN( PHI1)
      CPHI1 = COS(PHI1)
      X0 = RE*CPHI0*COS(LAMDA0)
      Y0 = RE*CPHI0*SIN(LAMDA0)
      Z0 = RE*SPHI0
      SL = SIN(LAMDA)
      CL = COS(LAMDA)
      SPHI = SIN(PHI)
      CPHI = COS(PHI)
      LONG = LAMDA0
      CALL LAUNCH
      VX = VT*COS(EL)*COS(AZ)
      VY = VT*COS(EL)*SIN(AZ)
      VZ =-VT*SIN(EL)
      IF(NSKIP .EQ. 0) GO TO 318
      R = RE + ALT
      LBAR = LAMDA + OMEGA*(TIME - T0)
      Z = R*SPHI
      Y = R*CPHI*SIN(LBAR)
      X = R*CPHI*COS(LBAR)
      CALL MATROC
      NLIN = -1
      GO TO 319
  318 CONTINUE
      CAZ = COS(AZ)
      SAZ = SIN(AZ)
      VXL = VX*CAZ + VY*SAZ
      VZL =    VZ
      XL = LENGTH*CEL
      ZL = ALT
      THETA = EL
      CALL ATMSPH
C
C         SOLVE ANGLE OF ATTACK EQUATIONS
```

```
      CALL ALFA
      CALL MATLCH
      NLCH = XL*CAZ
      ELCH = XL*SAZ
      ZLCH = ZL
      ZZ =  ZLCH - RE
      CALL TRANS2(NLCH,ELCH,ZZ,UX,UY,UZ)
      X = UX
      Y = UY
      Z = UZ
      R = SQRT(X**2 + Y**2 + Z**2)
      ALT = R - RE
      VX = VXL*CAZ
      VY = VXL*SAZ
      VZ =    VZL
  319 CONTINUE
C
C         COMPUTE INTIIAL VELOCITY
      CALL TRANS2(VX,VY,VZ,UX,UY,UZ)
      XDOT = -OMEGA*Y + UX
      YDOT =  OMEGA*X + UY
      ZDOT =  UZ
C
C     INITIALIZE S ARRAY
      N = 1
      S(1,N) = X
      S(2,N) = Y
      S(3,N) = Z
      S(4,N) = XDOT
      S(5,N) = YDOT
      S(6,N) = ZDOT
C
C         COMPUTE ACCELERATION
      NK = 1
      CALL ACCEL3
      THETA = EL
      ELPREV = EL
      IF(VT .LT. 1.0E-5) GO TO 320
C
C         COMPUTE OUTPUT DATA
      CALL DATA1
      NN = 2
C
C         STORE DATA TEMPORARILY
```

```
      CALL SETSA
      IF(NSKIP .EQ. 0) GO TO 320
      PTIME = TIME
      TFRAC = 0.0
      NLIPST = NLIP
      NLIP = 0
      CALL OUT
      NLIP = NLIPST
      TIM1 = AINT(TIME/STPTI)*STPTI + STPTI
      TIM2 = AINT(TIME/PTI)*PTI + PTI
      NLIN = NCYC -  INT(10.*(TIM1 - TIM2 + 1.0E-5)) - 1
  320 CONTINUE
      PTIME = AINT(TIME/PTI)*PTI + PTI
      TPREV = TIME
      N = 1
      ST(1,N) = X
      ST(2,N) = Y
      ST(3,N) = Z
      ST(4,N) = XDOT
      ST(5,N) = YDOT
      ST(6,N) = ZDOT
      ST(7,N) = XDDOT
      ST(8,N) = YDDOT
      ST(9,N) = ZDDOT
      IF(VT .GT. 1.0E-3) GO TO 350
      DT = LOW
      CALL RUNGE(ACCEL,3)
      DT = DTO
      GO TO 351
C
C          FIND PRESENT ROCKET MOTOR WEIGHT
  340 CONTINUE
      IF(ABS(1.0 - TSEP(NMOT)/TIME) .LT. 1.0E-6) GO TO 331
      GO TO 332
  331 WRM = WRM - WTM(NMOT)
      LSPT = LSPT + 1
      DO 33  J = 1,9
      SPS(J,LSPT) = ST(J,1)
  330 CONTINUE
      SPS(11,LSPT) = EL
      SPS(1 ,LSPT) = TIME
      NMOT = NMOT + 1
  332 CONTINUE
C
```

```
C         READ IN DATA FOR NEXT PHASE
      CALL TABLE
      KPH = KPH + KDEL
      NPH = NPH + NDEL
  350 CONTINUE
C
C         CALCULATE NEXT TRAJECTORY POINT AT T + DT
      CALL RUNGE(ACCEL1)
  351 CONTINUE
      TPDT = TIME + DT
      IF((TPDT-PTIME) .GT. LOW .AND. TIME .LT. PTIME) DT =
     1 PTIME - TIME + LOW
C
C         COMPUTE OUTPUT DATA
      CALL DATA1
      NN = 1
C
C         STORE DATA TEMPORARILY
      CALL SETSA
      IF(NSTOP .EQ. 0) GO TO 708
C
C         CHECK STOP TIME
      IF(TIME .GT. TSTOP) GO TO 703
      IF(ABS(1.0 - TIME/TSTOP) .LT. 1.0E-6) GO TO 701
      GO TO 708
  701 IEND = 1
      TFRAC = 1.0
      PTIME = TIME
      GO TO 801
  703 IEND = 1
      TFRAC = (TSTOP - TPREV)/(TIME - TPREV)
      NIIP = 0
      PTIME = TSTOP
      PHAS = TST
      GO TO 801
  708 CONTINUE
C
C         ALTITUDE=0 CHECK
      IF(ALT .LE. 0.0 .AND. EL .LT. 0.0) GO TO 710
      GO TO 720
  710 TFRAC = SA(5,2,1)/(SA(5,2,1) - SA(5,1,1))
      IEND = 1
      PTIME = TPREV + TFRAC*(TIME - TPREV)
      NIIP = 0
```

```
      PHAS = ALTO
      GO TO 801
  720 CONTINUE
      IF(NAP .EQ. 2) GO TO 721
C
C         APOGEE CHECK
      IF(FLPREV .LT. 0.0 .OR. FL .GT. 0.0) GO TO 721
      PHAS = APO
      KDFL = 1
      NDFL = 0
      IF(KSTOP .EQ. 0) GO TO 728
      IEND = 1
      TFRAC = 1.0
      PTIME = TIME
      GO TO 801
  728 CONTINUE
      NAP = 1
      TFRAC = 1.0
      PTIME = TIME
      NIIPST = NIIP
      NIIP = 0
      CALL OUT
      NIIP = NIIPST
      TIM1 = AINT(TIME/STPTI)*STPTI + STPTI
      TIM2 = AINT(TIME/PTI)*PTI + PTI
      NLIN = NCYC -  INT(10.*(TIM1 - TIM2 + 1.0E-5)) - 1
      PTIME = AINT(TIME/PTI)*PTI + PTI
      GO TO 805
  721 CONTINUE
C
C         IF NEXT PHASE HAS STARTED, PRINTOUT DATA
      IF(INEXT .NE. 1) GO TO 780
      INEXT = 0
      PTIME = TIME
      CALL OUT
      NIIP = NIIPST
      TIM1 = AINT(TIME/STPTI)*STPTI + STPTI
      TIM2 = AINT(TIME/PTI)*PTI + PTI
      NLIN = NCYC -  INT(10.*(TIM1 - TIM2 + 1.0E-5)) - 1
      DT = DTO
      PTIME = AINT(TIME/PTI)*PTI + PTI
      GO TO 805
  780 CONTINUE
C
```

```
C         AT END OF PHASE, PRINTOUT DATA
          IF(INEXT .NE. 2) GO TO 790
          IF(ABS(1.0 - PHT(NPH)/TIME) .GT. 1.0E-5) GO TO 790
          TFRAC = 1.0
          PTIME = TIME
          NIIPST = NIIP
          NIIP = 0
          CALL OUT
          INEXT = 4
          PTIME = AINT(TIME/PTI)*PTI + PTI
          GO TO 805
  790     CONTINUE
C
C         CHECK TIME FOR PRINTOUT
          IF(IEND .EQ. 1) GO TO 801
          IF(TIME .LT. PTIME) GO TO 805
          IF(TIME .EQ. PTIME) GO TO 800
C
C         COMPUTE INTERPOLATION FRACTION
          TFRAC = (PTIME - TPREV)/(TIME - TPREV)
          GO TO 801
  800     TFRAC = 1.0
          PTIME = TIME
  801     CONTINUE
C
C         CALL PRINTOUT ROUTINE
          CALL OUT
          IF(IEND .EQ. 1) GO TO 840
C
C         SET NEW PRINT TIME
          PTIME = PTIME + PTI
  805     CONTINUE
C
C         STORE PRESENT VALUES OF ALL OUTPUT DATA
          NN = 2
          CALL SETSA
          TPREV = TIME
          ELPREV = EL
  806     CONTINUE
C
C         PHASE TIME CHECK
          IF(NAP .EQ. 1) GO TO 810
          GO TO 811
  810     NAP = 2
```

```
      DT = 10.0*LOW
      GO TO (340,332),KCON
  811 CONTINUE
      IF(NPH .GT. NTAB) GO TO 350
      IF(ABS(1.0 - PHT(NPH)/TIME) .GT. 1.0E-5) GO TO 725
      KDEL = 1
      NDEL = 1
      INEXT = 1
      DT = 2.*LOW
      GO TO 340
  725 CONTINUE
      IF((TIME + DT) .GE. PHT(NPH)) GO TO 730
      IF(ABS(1.0 - PHT(NPH)/(TIME + DT)) .LT. 1.0E-5) GO TO 730
      GO TO 350
  730 CONTINUE
      IF(PHT(NPH) .LE. PTIME) GO TO 713
      IF((PHT(NPH) - PTIME) .LT. 4.*LOW) GO TO 713
      DT = PTIME - TIME + LOW
      GO TO 350
  713 CONTINUE
      DT = PHT(NPH) - TIME
      INEXT = 2
      GO TO 350
C
C
C         SPENT STAGE ROUTINE
  840 NPHS = 0
      NPH = 1
      KCON = 2
      IF(NSPENT)999,999,850
  850 KSPENT = KSPENT + 1
      KPH = 1
      KDEL = 1
      KPAG = 0
      DO 65  J = 1,14
      PHT(J) = 900000.
  650 CONTINUE
      DO 855 J = 1,9
      IF(NPAGE(J) .EQ. 5) GO TO 856
  855 CONTINUE
      GO TO 858
  856 DO 857 I = J,9
      NPAGE(I) = NPAGE(I+1)
  857 CONTINUE
```

[redacted]

```
  858 CONTINUE
      DO 851 J = 1,9
      ST(J,1) = SPS(J,KSPENT)
      S(J,1) = ST(J,1)
  851 CONTINUE
      TIME = SPS(10,KSPENT)
      ELPREV = SPS(11,KSPENT)
      SA(5,2,1) = 0.0
      NITP = 0
      IEND = 0
      KSTOP = 0
      NSTOP = 0
      NAP =
      THCON = 1
      DT = 10.0*LOW
      TPREV = TIME
      NSPENT = NSPENT - 1
      READ(5,NAM5)
      WTM(KSPENT) = WTS
      ARG = TIME/PTI
      PTIME = AINT(ARG)*PTI + PTI
      GO TO 332
  999 STOP
      END
```

```
CACCELO     SUB. ACCELO --- ALFA ACCELERATION EQS.
      SUBROUTINE ACCELO
      DOUBLE PRECISION TMAT,SMAT
      COMMON/STOR1/ALT,TEMP,PRES,DENS,VISC,SOUND
      COMMON/STOR2/X,Y,Z,R
      COMMON/STOR3/TMAT,SMAT,SPHI,CPHI,CL,SL
      COMMON/STOR4/SAREA,MCH,CDD,CD,MACH,NCD,CDS,DRAG
      COMMON/STOR5/TWT,NWT,WPR,WEIGHT,MASS
      COMMON/STOR6/PO,WPM,WPL,WPP2,WTM
      COMMON/STOR7/XDOT,YDOT,ZDOT,XDDOT,YDDOT,ZDDOT
      COMMON/STOR8/OMEGA,RE,RA,RB,GO,GM,ALIM
      COMMON/STOR9/S,NK
      COMMON/STOR13/XL,ZL,THETA,VXL,VZL,ALPH,GAMA,CAZ,SAZ
      COMMON/STOR14/LSM,INERT,CNA
      COMMON/STOR15/Q,VX,VY,VZ,VT,EL,AZ,RANGE,LAT,LONG,ACC,GDLAT
      COMMON/STOR19/TIME,TO
      COMMON/STOR20/AREA,IOPT,TIM,NTH,THS,THRUST
      COMMON/STOR23/SPHIO,CPHIO,LAMDAO,XO,YO,ZO
      COMMON/STOR25/LOW,EPTINY,EPBIG,PTI,PI,PI2,DT,HIGH
      DIMENSION TIM(50),THS(50),TWT(50),WPR(50),MCH(50),CDD(50)
      DIMENSION SMAT(3,3),TMAT(3,3),WTM(6),S(9,6)
      REAL LAT,LONG,LAMDAO,LOW
      REAL LSM,INERT,MASS,MACH,MCH
      N = NK
      XL = S(1,N)
      ZL = S(2,N)
      THETA = S(3,N)
      VXL = S(4,N)
      VZL = S(5,N)
      VL = SQRT(VXL**2 + VZL**2)
      CALL ATMSPH
      MACH = VL/SOUND
      Q = (DENS*VL**2)/2.
C        COMPUTE DRAG
      CALL TAB1
      DRAG = CDS*Q
C        COMPUTE THRUST
      CALL TAB2
C        COMPUTE WEIGHT
      CALL TAB3
      MASS = WEIGHT/GO
```

```
      GAMA = ATAN2(VZL,VXL)
      ALPH = THETA - GAMA
      CS = COS(THETA)
      SN = SIN(THETA)
      FNORM =  CNA*SAREA*Q*ALPH
      AXL = (THRUST*CS - DRAG*CS    - FNORM*SN)/MASS
      AZL = (THRUST*SN - DRAG*SN    + FNORM*CS)/MASS - GO
      THDD = (-LSM*FNORM)/INERT
      S(7,N) = AXL
      S(8,N) = AZL
      S(9,N) = THDD
      RETURN
      END
```

O.C.

```
CACCEL1     SUB. ACCEL1 --- ACCELERATION FOR RUNGE
            SUBROUTINE ACCEL1
            DOUBLE PRECISION TMAT,SMAT
            COMMON/STOR1/ALT,TEMP,PRES,DENS,VISC,SOUND
            COMMON/STOR2/X,Y,Z,R
            COMMON/STOR3/TMAT,SMAT,SPHI,CPHI,CL,SL
            COMMON/STOR4/SAREA,MCH,CDD,CD,MACH,NCD,CDS,DRAG
            COMMON/STOR5/TWT,NWT,WPR,WEIGHT,MASS
            COMMON/STOR6/P0,WRM,WPL,WPP2,WTM
            COMMON/STOR7/DXDT,DYDT,DZDT,XDDOT,YDDOT,ZDDOT
            COMMON/STOR8/OMEGA,RE,RA,RB,GO,GM,ALIM
            COMMON/STOR9/S,NK
            COMMON/STOR11/THCON,T1,ALTC,THX,THY,THZ
            COMMON/STOR15/Q,VX,VY,VZ,VT,EL,AZ,RANGE,LAT,LONG,ACC,GDLAT
            COMMON/STOR19/TIME,TO
            COMMON/STOR20/AREA,IOPT,TIM,NTH,THS,THRUST
            COMMON/STOR26/ KCON,KSPENT
            DIMENSION TIM(50),THS(50),TWT(50),WPR(50),MCH(50),CDD(50),WTM(6)
            DIMENSION SMAT(3,3),TMAT(3,3),S(9,6)
            INTEGER THCON
            REAL LAMDAO,MASS ,LOW,LONG, LAT, MACH,MCH
            N = NK
C               INPUT PRESENT POSITION AND VELOCITY
            X = S(1,N)
            Y = S(2,N)
            Z = S(3,N)
            DXDT = S(4,N)
            DYDT = S(5,N)
            CXDT = DXDT + OMEGA*Y
            CYDT = DYDT - OMEGA*X
            DZDT = S(6,N)
            CALL MATROC                               SET UP ROTATION FOR PRESENT POSITION —
            CALL TRANS1(CXDT,CYDT,DZDT,WX,WY,WZ)      INERTIAL TO ~~EARTH FIXED~~ TOPOCENTRIC
            VX = WX                                   (VELOCITY)
            VY = WY
            VZ = WZ
            VT = SQRT(VX**2 + VY**2 + VZ**2)
            R = SQRT(X**2 + Y**2 + Z**2)
            ALT = R - RE
            IF(ALT .GE. ALIM) GO TO 80
C               CALL ATMSPH TO FIND SPEED OF SOUND, DENSITY, AND PRESSURE
```

```
      CALL ATMSPH
      MACH = VT/SOUND
      GO TO 90
   80 CONTINUE
      MACH = -2.0
      PRES = 0.0
      DENS = 0.0
   90 CONTINUE
      Q = (DENS*VT**2)/2.
C        COMPUTE DRAG
C        COMPUTE THRUST
C        COMPUTE TOTAL WEIGHT
      GO TO (95,98),KCON
   95 CALL TAB1
      CALL TAB2
      CALL TAB3
      GO TO 100
   98 CALL TAB1
      THRUST = 0.0
      WEIGHT = WTM(KSPENT)
  100 CONTINUE
      DRAG = CDS*Q
      MASS = WEIGHT/G0
C        COMPUTE GRAVITATIONAL ACCELERATION
      GG = GM/R**3
      GX = -GG*X
      GY = -GG*Y
      GZ = -GG*Z
      FX = DRAG*VX/VT
      FY = DRAG*VY/VT
      FZ = DRAG*VZ/VT
      CALL TRANS2(FX,FY,FZ,UX,UY,UZ)
      CALL DIRECT
C        COMPUTE TOTAL INERTIAL ACCELERATION COMPONENTS
      XDDOT = GX + (THX - UX)/MASS
      YDDOT = GY + (THY - UY)/MASS
      ZDDOT = GZ + (THZ - UZ)/MASS
      S(7,N) = XDDOT
      S(8,N) = YDDOT
      S(9,N) = ZDDOT
      RETURN
      END
```

ROT TO INERT.
THRUST CONTROL.

O.C.

```
CACCEL2      SUB. ACCEL2 --- LAUNCH RAIL ACCELERATION
      SUBROUTINE ACCEL2
      DOUBLE PRECISION ST
      COMMON/STOR1/ALT,TEMP,PRES,DENS,VISC,SOUND
      COMMON/STOR4/SAREA,MCH,CDD,CD,MACH,NCD,CDS,DRAG
      COMMON/STOR5/TWT,NWT,WPR,WEIGHT,MASS
      COMMON/STOR8/OMEGA,RE,RA,RB,GO,GM,ALIM
      COMMON/STOR9/S,NK
      COMMON/STOR10/ST
      COMMON/STOR15/Q,VX,VY,VZ,VT,EL,AZ,RANGE,LAT,LONG,ACC,GDLAT
      COMMON/STOR17/LENGTH,SEL,DIST,LVEL,LACC
      COMMON/STOR19/TIME,TO
      COMMON/STOR20/AREA,IOPT,TIM,NTH,THS,THRUST
      DIMENSION ST(10,10)
      DIMENSION MCH(50),CDD(50),TWT(50),WPR(50),TIM(50),THS(50),S(9,6)
      REAL LVEL,LACC,MACH,MASS
      N = NK
      DIST = S(1,N)
      LVEL = S(4,N)
      MACH = LVEL/SOUND
      Q = (DENS*LVEL**2)/2.
C        COMPUTE DRAG
      CALL TAB1
      DRAG = CDS*Q
C        COMPUTE THRUST
      CALL TAB2
C        COMPUTE WEIGHT
      CALL TAB3
      MASS = WEIGHT/GO
      LACC = (THRUST - WEIGHT*SEL - DRAG)/MASS
      S(7,N) = LACC
      RETURN
      END
```

```
CACCEL3      SUB. ACCEL3 --- ACCELERATION USING EL,AZ
      SUBROUTINE ACCEL3
      DOUBLE PRECISION TMAT,SMAT
      COMMON/STOR1/ALT,TEMP,PRES,DENS,VISC,SOUND
      COMMON/STOR2/X,Y,Z,R
      COMMON/STOR3/TMAT,SMAT,SPHI,CPHI,CL,SL
      COMMON/STOR4/SAREA,MCH,CDD,CD,MACH,NCD,CDS,DRAG
      COMMON/STOR5/TWT,NWT,WPR,WEIGHT,MASS
      COMMON/STOR7/XDOT,YDOT,ZDOT,XDDOT,YDDOT,ZDDOT
      COMMON/STOR8/OMEGA,RE,RA,RB,GO,GM,ALIM
      COMMON/STOR9/S,NK
      COMMON/STOR15/Q,VX,VY,VZ,VT,EL,AZ,RANGE,LAT,LONG,ACC,GDLAT
      COMMON/STOR20/AREA,IOPT,TIM,NTH,THS,THRUST
      DIMENSION SMAT(3,3),TMAT(3,3),S(9,6)
      DIMENSION TIM(50),THS(50),TWT(50),WPR(50),MCH(50),CDD(50)
      REAL MASS,LONG,LAT,MACH,MCH
      N = NK
C        INPUT PRESENT POSITION AND VELOCITY
      X = S(1,N)
      Y = S(2,N)
      Z = S(3,N)
      DXDT = S(4,N)
      DYDT = S(5,N)
      CXDT = DXDT + OMEGA*Y
      CYDT = DYDT - OMEGA*X
      DZDT = S(6,N)
      CALL MATROC
      CALL TRANS1(CXDT,CYDT,DZDT,WX,WY,WZ)
      VX = WX
      VY = WY
      VZ = WZ
      VT = SQRT(VX**2 + VY**2 + VZ**2)
      CELCAZ = COS(EL)*COS(AZ)
      CELSAZ = COS(EL)*SIN(AZ)
      SEL = SIN(EL)
      IF(VT .LT. 1.0E-5) GO TO 70
      CELCAZ = VX/VT
      CELSAZ = VY/VT
      SEL =-VZ/VT
   70 CONTINUE
      R = SQRT(X**2 + Y**2 + Z**2)
```

```
      ALT = R - RE
      IF(ALT .GE. ALIM) GO TO 80
C        CALL ATMSPH TO FIND SPEED OF SOUND, DENSITY, AND PRESSURE
      CALL ATMSPH
      MACH = VT/SOUND
      GO TO 90
   80 CONTINUE
      MACH = -2.0
      PRES = 0.0
      DENS = 0.0
   90 CONTINUE
      Q = (DENS*VT**2)/2.
      CALL TAB1
      DRAG = CDS*Q
C        COMPUTE THRUST
      CALL TAB2
C        COMPUTE ROCKET WEIGHT
      CALL TAB3
      MASS = WEIGHT/G0
      DIFF = THRUST - DRAG
C        COMPUTE GRAVITATION ACCELERATION
      GG = GM/R**3
      GX = -GG*X
      GY = -GG*Y
      GZ = -GG*Z
C        COMPUTE AERO FORCES IN TOPOCENTRIC SYSTEM
      FX = DIFF*CELCAZ
      FY = DIFF*CELSAZ
      FZ =-DIFF*SEL
C        TRANSFORM AERO FORCES TO INERTIAL SYSTEM
      CALL TRANS2(FX,FY,FZ,UX,UY,UZ)
C        COMPUTE INERTIAL ACCELERATION
      XDDOT = GX + UX/MASS
      YDDOT = GY + UY/MASS
      ZDDOT = GZ + UZ/MASS
      S(7,N) = XDDOT
      S(8,N) = YDDOT
      S(9,N) = ZDDOT
      RETURN
      END
```

```
CALFA         SUB. ALFA --- ANGLE OF ATTACK EQUATIONS
      SUBROUTINE ALFA
      DOUBLE PRECISION ST
      DOUBLE PRECISION DP1
      COMMON/STOR1/ALT,TEMP,PRES,DENS,VISC,SOUND
      COMMON/STOR2/X,Y,Z,R
      COMMON/STOR4/SAREA,MCH,CDD,CD,MACH,NCD,CDS,DRAG
      COMMON/STOR5/TWT,NWT,WPR,WEIGHT,MASS
      COMMON/STOR7/XDOT,YDOT,ZDOT,XDDOT,YDDOT,ZDDOT
      COMMON/STOR8/OMEGA,RE,RA,RB,G0,GM,ALIM
      COMMON/STOR9/S,NK
      COMMON/STOR10/ST
      COMMON/STOR13/XL,ZL,THETA,VXL,VZL,ALPH,GAMA,CAZ,SAZ
      COMMON/STOR14/LSM,INERT,CNA
      COMMON/STOR15/Q,VX,VY,VZ,VT,EL,AZ,RANGE,LAT,LONG,ACC,GDLAT
      COMMON/STOR16/SA,ARRAY,NN
      COMMON/STOR18/TFRAC,LINE,PTIME,CONV,IEND,IPRINT,NPAGE
      COMMON/STOR19/TIME,T0
      COMMON/STOR20/AREA,IOPT,TIM,NTH,THS,THRUST
      COMMON/STOR23/SPHI0,CPHI0,LAMDA0,X0,Y0,Z0
      COMMON/STOR24/PHT,NID,NTAB,NPH,KPH
      COMMON/STOR25/LOW,EPTINY,EPBIG,PTI,PI,PI2,DT,HIGH
      DIMENSION TIM(50),THS(50),S(9,6),TWT(50),WPR(50)
      DIMENSION NPAGE(10)
      DIMENSION PHT(14),NID(14),MCH(50),CDD(50)
      DIMENSION ST(20,10),ARRAY(10,30,10),SA(10,2,10)
      REAL LAT,LONG,LAMDA0,LOW
      REAL MCH,MACH
      REAL LSM,INERT,MASS
      EXTERNAL ACCEL0
      PTIME = AINT(TIME/PTI)*PTI + PTI
      TLIM = PHT(2) - 1.0
      TPREV = TIME
      CALL ATMSPH
      MACH = VT/SOUND
      Q = (DENS*VT**2)/2.
      CALL TAB1
      DRAG = CDS*Q
      CALL TAB2
      CALL TAB3
      MASS = WEIGHT/G0
```

```
C          INITIALIZE LAUNCH VARIABLES
      AXL = (THRUST - DRAG)*COS(THETA)/MASS
      AZL = (THRUST - DRAG)*SIN(THETA)/MASS - G0
      THDD = 0.0
      IF(VT .GT. 1.0E-5) GO TO 80
      DP1 = VXL
      VXL = DP1 + (1.0D-6)*AXL
      DP1 = VZL
      VZL = DP1 + (1.0D-6)*AZL
      TIME = TIME + 1.0E-6
   80 CONTINUE
      VL = SQRT(VXL**2 + VZL**2)
C          INITIALIZE S ARRAY
      N = 1
      S(1,N) = XL
      S(2,N) = ZL
      S(3,N) = THETA
      S(4,N) = VXL
      S(5,N) = VZL
      S(6,N) = 0.0
      S(7,N) = AXL
      S(8,N) = AZL
      S(9,N) = THDD
      DO 90 J = 1,6
      ST(J,1) = S(J,1)
   90 CONTINUE
  100 CONTINUE
      CALL RUNGE(ACCEL0)
      TPDT = TIME + DT
      IF((TPDT - PTIME) .GT. LOW .AND. TIME .LT. PTIME) DT =
     1 PTIME - TIME + LOW
      AXL = S(7,1)
      AZL = S(8,1)
      VT = SQRT(VXL**2 + VZL**2)
      CALL MATLCH
      XT = XL*CAZ
      YT = XL*SAZ
      ZT = - ZL
      ZZT = ZT - RE
      CALL TRANS2(XT,YT,ZZT,UX,UY,UZ)
```

O.C.

```
      X = UX
      Y = UY
      Z = UZ
      R = SQRT(X**2 + Y**2 + Z**2)
      ALT = R - RE
      VX = VXL*CAZ
      VY = VXL*SAZ
      VZ = - VZL
      CALL TRANS2(VX,VY,VZ,UX,UY,UZ)
      XDOT = -OMEGA*Y + UX
      YDOT =  OMEGA*X + UY
      ZDOT =  UZ
      ACX = AXL*CAZ
      ACY = AXL*SAZ
      ACZ = -AZL
      CALL TRANS2(ACX,ACY,ACZ,UX,UY,UZ)
      XDDOT = UX - 2.*OMEGA*YDOT + OMEGA**2*X
      YDDOT = UY + 2.*OMEGA*XDOT + OMEGA**2*Y
      ZDDOT = UZ
      CALL DATA1
      NN = 1
      CALL SETSA
      IF(ALPH .GT. 0.0) GO TO 750
      TFRAC = SA(1,2,1)/(SA(1,2,1) - SA(1,1,1))
      TFRAC = ABS(TFRAC)
      TIME = TPREV + TFRAC*(TIME - TPREV)
      XL = XL2 + TFRAC*(XL - XL2)
      ZL = ZL2 + TFRAC*(ZL - ZL2)
      VXL = VXL2 + TFRAC*(VXL - VXL2)
      VZL = VZL2 + TFRAC*(VZL - VZL2)
      GAMA = ATAN2(VZL,VXL)
      EL = GAMA
      GO TO 200
  750 CONTINUE
      IF(TIME .LT. PTIME) GO TO 805
      IF(TIME .EQ. PTIME) GO TO 800
C        COMPUTE INTERPOLATION FRACTION
      TFRAC = (PTIME - TPREV)/(TIME - TPREV)
      GO TO 801
  800 TFRAC = 1.0
```

```
      PTIME = TIME
  801 CONTINUE
C         CALL PRINTOUT ROUTINE
      CALL OUT
C         SET NEW PRINT TIME
      PTIME = PTIME + PTI
  805 CONTINUE
C         STORE PRESENT VALUES OF ALL OUTPUT DATA
      NN = 2
      CALL SETSA
      TPREV = TIME
      XL2 = XL
      ZL2 = ZL
      VXL2 = VXL
      VZL2 = VZL
      IF(TIME .GT. 20.0) GO TO 200
      IF(TIME .LT. TLIM) GO TO 100
  200 RETURN
      END
```

[redacted]

```
3ATMSPH     SUB. ATMSPH --- ATMOSPHERIC VARIABLES
      SUBROUTINE ATMSPH
      COMMON/STOR1/ALT,TEMP,PRES,DENS,VISC,SOUND
      COMMON/STOR8/OMEGA,RE,RA,RB,G0,GM,ALIM
      COMMON/STOR36/PRES0,DENS0,TEMP0,SOUND0,K1,CON1,KPREV
      COMMON/STOR37/TSLOPE,HALTB,TEMPB,PRESB
      DIMENSION TSLOPE( 9),HALTB( 9),TEMPB( 9),PRESB( 9)
      REAL K1
      HALT = ALT*(RE/(RE + ALT))*CON1
      CALL INTERP(HALT,N,$999,$800)
      DHALT = HALT - HALTB(N)
      TEMP = TEMPB(N) + TSLOPE(N)*DHALT
      IF(TSLOPE(N))400,500,400
  400 CONTINUE
      PRATIO = PRESB(N)*(TEMPB(N)/TEMP)**(K1/TSLOPE(N))
      GO TO 600
  500 CONTINUE
      PRATIO = PRESB(N)*EXP((-K1/TEMPB(N))*DHALT)
  600 CONTINUE
      TRATIO = TEMP/TEMP0
      DENS = DENS0*PRATIO/TRATIO
      SOUND = SOUND0*SQRT(TRATIO)
      PRES = PRATIO*PRES0
      RETURN
  800 PRES = 0.0
      DENS = 0.0
      DENS = 0.0
      SOUND = 0.0
      RETURN
  999 PRES = PRES0
      DENS = DENS0
      TEMP = TEMP0
      SOUND = SOUND0
      RETURN
      END
```

```
CBLOCK
      BLOCK DATA
      COMMON/STOR36/PRESO,DENSO,TEMPO,SOUNDO,K1,CON1,KPREV
      COMMON/STOR37/TSLOPE,HALTB,TEMPB,PRESB
      DIMENSION TSLOPE( 9),HALTB( 9),TEMPB( 9),PRESB( 9)
      REAL K1
      DATA TSLOPE/ -6.5E-3,0.0,1.0E-3,2.8E-3,0.0,-2.0E-3,-4.0E-3,0.0,
     1 0.0/
      DATA HALTB/ 0.0,11.0E+3,20.0E+3,32.0E+3,47.0E+3,52.0E+3,61.0E+3,
     1  79. E+3,88.743E+3/
      DATA TEMPB/ 288.15,216.65,216.65,228.65,270.65,270.65,252.65,
     1 180.65,180.65/
      DATA PRESB/ 1.0,2.23361E-1,5.40328E-2,8.56663E-3,1.09455E-3,
     1 5.82289E-4,1.79718E-4,1.0241E-5,1.6223E-6/
      DATA PRESO/2116.2/,DENSO/2.3769E-3/,TEMPO/288.15/,
     1 SOUNDO/1116.45/,K1/34.163194E-3/,CON1/.3048/,KPREV/1/
      END
```

```
CDATA1      SUB. DATA1 --- COMPUTE ADDITIONAL OUTPUT
      SUBROUTINE DATA1
      DOUBLE PRECISION CCS
      COMMON/STOR1/ALT,TEMP,PRES,DENS,VISC,SOUND
      COMMON/STOR2/X,Y,Z,R
      COMMON/STOR7/XDOT,YDOT,ZDOT,XDDOT,YDDOT,ZDDOT
      COMMON/STOR8/OMEGA,RE,RA,RB,GO,GM,ALIM
      COMMON/STOR12/YL,VYL,RXY,GAML,RLDOT
      COMMON/STOR13/XL,ZL,THETA,VXL,VZL,ALPH,GAMA,CAZ,SAZ
      COMMON/STOR15/Q,VX,VY,VZ,VT,EL,AZ,RANGE,LAT,LONG,ACC,GDLAT
      COMMON/STOR19/TIME,TO
      COMMON/STOR25/LOW,EPTINY,EPBIG,PTI,PI,PI2,DT,HIGH
      COMMON/STOR27/IIPLAT,IIPLON,IIPR,IIPTIM
      COMMON/STOR28/RLCH,AZLCH,ELLCH,VELLCH,ELCH,NLCH,ZLCH,EDLCH,
     1 NDLCH,ZDLCH
      COMMON/STOR29/RRAD,AZRAD,ELRAD,VELRAD,ERAD,NRAD,ZRAD,EDRAD,
     1 NDRAD,ZDRAD
      COMMON/STOR30/DELTA,VEL,ACCI
      REAL NRAD,NLCH,NDRAD,NDLCH
      REAL IIPLAT,IIPLON,IIPR,IIPTIM
      REAL LAMDA,MASS,K,LOW,LONG, LAT, MACH,MCH,LAMDAO
      VEL = SQRT(XDOT**2 + YDOT**2 + ZDOT**2)
      ACCI = SQRT(XDDOT**2 + YDDOT**2 + ZDDOT**2)
      EX = XDDOT + 2.*OMEGA*YDOT - OMEGA**2*X
      EY = YDDOT - 2.*OMEGA*XDOT - OMEGA**2*Y
      EZ = ZDDOT
      CALL TRANS1(EX,EY,EZ,ACX,ACY,ACZ)
      ACC = SQRT(ACX**2 + ACY**2 + ACZ**2)
      CALL MATRAD
      CALL TRANS1(X,Y,Z,WX,WY,WZ)
      ZRAD =-WZ - RE
      ERAD = WY
      NRAD = WX
      RRAD = SQRT(ERAD**2 + NRAD**2 + ZRAD**2)
      CXDT = XDOT + OMEGA*Y
      CYDT = YDOT - OMEGA*X
      CZDT = ZDOT
      CALL TRANS1(CXDT,CYDT,CZDT,WX,WY,WZ)
      EDRAD = WY
      NDRAD = WX
      ZDRAD = -WZ
```

```
      VELRAD = SQRT(EDRAD**2 +NDRAD**2 + ZDRAD**2)
      SRT = SQRT(ERAD**2 + NRAD**2)
      ELRAD = ATAN2(ZRAD,SRT)
      AZRAD = ATAN2(ERAD,NRAD)
      IF(AZRAD .LT. 0.0) AZRAD = AZRAD + PI2
      CALL MATLCH
      CALL TRANS1(X,Y,Z,WX,WY,WZ)
      ELCH = WY
      NLCH = WX
      ZLCH = -WZ - RE
      RLCH = SQRT(ELCH**2 + NLCH**2 + ZLCH**2)
      CALL TRANS1(CXDT,CYDT,CZDT,WX,WY,WZ)
      EDLCH = WY
      ZDLCH = -WZ
      NDLCH = WX
      VELLCH = SQRT(EDLCH**2 + NDLCH**2 + ZDLCH**2)
      SRT = SQRT(ELCH**2 + NLCH**2)
      ELLCH = ATAN2(ZLCH,SRT)
      AZLCH = ATAN2(ELCH,NLCH)
      IF(AZLCH .LT. 0.0) AZLCH = AZLCH + PI2
      XL = ELCH*SAZ + NLCH*CAZ
      YL =-ELCH*CAZ + NLCH*SAZ
      ZL = ZLCH
      VXL = EDLCH*SAZ + NDLCH*CAZ
      VYL =-EDLCH*CAZ + NDLCH*SAZ
      VZL = ZDLCH
      RXY = SQRT(XL**2 + YL**2)
      SRT = SQRT(VXL**2 + VYL**2)
      GAML = ATAN2(VZL,SRT)
      RLDOT = (XL*VXL + YL*VYL + ZL*VZL)/RLCH
      RANGE = 0.
      IF(RLCH .LT. 5.0) GO TO 700
      CCS = (R**2 + RE**2 - RLCH**2)/(2.*R*RE)
      IF(DABS(CCS) .GT. 1.0D0) CCS = 1.0D0
      DELTA = DATAN2(DSQRT(1.-CCS**2),CCS)
      RANGE = RE*DELTA
  700 CONTINUE
      SRT = SQRT(X**2 + Y**2)
      PHI = ATAN(Z/SRT)
      ARG = (RA/RB)**2*(Z/SRT)
```

o.c.

```
GDLAT = ATAN(ARG)
LAT = PHI
LAMDA = ATAN2(Y,X)
LONG = LAMDA - OMEGA*(TIME - TO)
IF(LONG .GT. PI) LONG = LONG - PI2
IF(LONG .LT. (-PI)) LONG = LONG + PI2
EL = ATAN(-VZ/SQRT(VX**2 + VY**2))
ALPH = THETA - EL
AZ = ATAN2(VY,VX)
IF(AZ .LT. 0.) AZ = AZ + PI2
IF(ALT .LT. 5.0 .AND. EL .LT. 0.0) RETURN
CALL IIP
RETURN
END
```

```
CDIRECT      SUB. DIRECT --- THRUST CONTROL OPTION
            SUBROUTINE DIRECT
            COMMON/STOR1/ALT,TEMP,PRES,DENS,VISC,SOUND
            COMMON/STOR11/THCON,T1,ALTC,THX,THY,THZ
            COMMON/STOR13/XL,ZL,THETA,VXL,VZL,ALPH,GAMA,CAZ,SAZ
            COMMON/STOR15/Q,VX,VY,VZ,VT,EL,AZ,RANGE,LAT,LONG,ACC,GDLAT
            COMMON/STOR19/TIME,TO
            COMMON/STOR20/AREA,IOPT,TIM,NTH,THS,THRUST
            DIMENSION TIM(50),THS(50)
            INTEGER THCON
            EL = ATAN(-VZ/SQRT(VX**2 + VY**2))
            XFRAC = VX/VT
            YFRAC = VY/VT
            ZFRAC = VZ/VT
            CALL TRANS2(XFRAC,YFRAC,ZFRAC,UX,UY,UZ)
            XFRAC = UX
            YFRAC = UY
            ZFRAC = UZ
            THETA = EL
            IF(TIME .LT. T1 .OR. NSET .EQ. 1 .OR. ALT .LT. ALTC) GO TO 100
            XFRAC1 = XFRAC
            YFRAC1 = YFRAC
            ZFRAC1 = ZFRAC
            ELST = EL
            TIMST = TIME
            NSET = 1
        100 CONTINUE
            IF(TIME .LT. T1 .OR. ALT .LT. ALTC) GO TO 201
            GO TO (201,202),THCON
        201 CONTINUE
C               NO THRUST CONTROL
            THX = THRUST*XFRAC
            THY = THRUST*YFRAC
            THZ = THRUST*ZFRAC
            GO TO 900
        202 CONTINUE
C               THRUST CONTROL OPTION
            THX = THRUST*XFRAC1
            THY = THRUST*YFRAC1
            THZ = THRUST*ZFRAC1
            CALL TRANS1(XFRAC1,YFRAC1,ZFRAC1,WX,WY,WZ)
            THETA = ATAN(-WZ/SQRT(WX**2 + WY**2))
        900 RETURN
            END
```

```
CIIP          SUB. IIP --- INSTANTANEOUS IMPACT POINTS
      SUBROUTINE IIP
      DOUBLE PRECISION SSIG2,CSIG2,SIG,SQ
      DOUBLE PRECISION ECE,ESE,ECEIP,ESEIP,SDEL,CDEL,DEL,F,G,A
      DOUBLE PRECISION XIP,YIP,ZIP,LC,ESQ,X0,Y0,Z0,X,Y,Z,R
      COMMON/STOR2/ XX,YY,ZZ,RR
      COMMON/STOR7/XDOT,YDOT,ZDOT,XDDOT,YDDOT,ZDDOT
      COMMON/STOR8/OMEGA,RE,PA,PB,G0,GM,ALIM
      COMMON/STOR13/XL,ZL,THETA,VX_,VZL,ALPH,GAMA,CAZ,SAZ
      COMMON/STOR19/TIME,TO
      COMMON/STOR23/SPHI0,CPHI0,LAMDA0,XX0,YY0,ZZ0
      COMMON/STOR25/LOW,EPTINY,EPBIG,PII,PI,PI2,DT,HIGH
      COMMON/STOR27/IIPLAT,IIPLON,IIPR,IIPTIM
      REAL LAMDA,MASS,K,LOW,LONG, LAT, MACH,MCH,LAMDA0,LBAR
      REAL IIPLAT,IIPLON,IIPR,IIPTIM
C        INITIALIZE  X,Y,Z,XDOT,YDOT,ZDOT,R,VEL
      X = XX
      Y = YY
      Z = ZZ
      R = RR
      VSQ = XDOT**2 + YDOT**2 + ZDOT**2
      SRGM = SQRT(GM)
      ECE = R*VSQ/GM - 1.0
      A = R/(1.0 - ECE)
      ESE = (X*XDOT + Y*YDOT + Z*ZDOT)/(SRGM*DSQRT(A))
      ESQ = ECE**2 + ESE**2
      ECEIP = 1.0D0 - RE/A
      SQ = ESQ - ECEIP**2
      IF(SQ .GT. 0.0D0) GO TO 100
      SQ = 0.0D0
  100 ESEIP = -DSQRT(SQ)
      SDEL = (ESEIP*ECE - ECEIP*ESE)/ESQ
      CDEL = (ECEIP*ECE + ESEIP*ESE)/ESQ
      DEL = DATAN2(SDEL,CDEL)
      IF(DEL .LT. 0.0D0) DEL = DEL + PI2
      F = (CDEL - ECE)/(1.0 - ECE)
      G = (A**1.5/SRGM)*(SDEL + ESE - ESEIP)
      TFL  = (A**1.5/SRGM)*(DEL + ESE - ESEIP)
      CAPT = TFL + (TIME - TO)
      XIP = F*X + G*XDOT
      YIP = F*Y + G*YDOT
```

```
      ZIP = F*Z + G*ZDOT
      LBAR = LAMDA0 + OMEGA*CAPT
      X0 = RE*CPHI0*COS(LBAR)
      Y0 = RE*CPHI0*SIN(LBAR)
      Z0 = RE*SPHI0
      LC =DSQRT((XIP - X0)**2 + (YIP - Y0)**2 + (ZIP -Z0)**2)
      LAMDA = ATAN2(YIP,XIP)
      LONG = LAMDA - OMEGA*CAPT
      PHI = ATAN(ZIP/SQRT(XIP**2 + YIP**2))
      GDPHI = ATAN((RA/RB)**2*SIN(PHI)/COS(PHI))
      SSIG2 = LC/(2.*RE)
      IF(DABS(SSIG2) .GT. 1.0D0) SSIG2 = 1.0D0
      CSIG2 = DSQRT(1.0 - SSIG2**2)
      SIG = 2.*DATAN(SSIG2/CSIG2)
      RANGE = RE*SIG
      IIPLAT = GDPHI
      IIPLON = LONG
      IIPR = RANGE
      IIPTIM = CAPT
      RETURN
      END
```

```
INPUT         SUB. INPUT --- READ INPUT AND PRINTOUT
      SUBROUTINE INPUT
      DIMENSION AA(14)
      DATA EN/6HZ99999/
      REWIND 5
   23 CONTINUE
      READ(5,19) AA
   19 FORMAT(14A6)
      PRINT 19, AA
      IF(AA(2) .EQ. EN) GO TO 34
      GO TO 23
   34 REWIND 5
      RETURN
      END
```

```
CINTERP
      SUBROUTINE INTERP(QQ,KP,*,*)
      COMMON/STOR36/PRESO,DENSO,TEMPO,SOUNDO,K1,CON1,KPREV
      COMMON/STOR37/TSLOPE,HALTB,TEMPB,PRESB
      DIMENSION TSLOPE( 9),HALTB( 9),TEMPB( 9),PRESB( 9)
      KP = KPREV
      IF(QQ .LE. HALTB(KP+1) .AND. QQ .GT. HALTB(KP)) GO TO 200
      IF(QQ .LE. HALTB(1)) RETURN1
      IF(QQ .GE. HALTB(9)) RETURN2
      IF(QQ   HPREV) 150,200,100
  100 DO 11  J=KP,8
      IF(QQ .GT. HALTB(J) .AND. QQ .LE. HALTB(J+1)) GO TO 120
  110 CONTINUE
  120 KP = J
      GO TO 200
  150 KPP = KP - 1
      DO 16  J=1,KPP
      KPJ = KP - J
      IF(QQ .GT. HALTB(KPJ ) .AND. QQ .LE. HALTB(KPJ+1))  GO TO 180
  160 CONTINUE
  180 KP = KP - J
  200 CONTINUE
      KPREV = KP
      HPREV = QQ
      RETURN
      END
```

```
CLAUNCH      SUB. LAUNCH --- LAUNCH RAIL
      SUBROUTINE LAUNCH
      DOUBLE PRECISION ST
      COMMON/STOR1/ALT,TEMP,PRES,DENS,VISC,SOUND
      COMMON/STOR4/SAREA,MCH,CDD,CL,MACH,NCD,CDS,DRAG
      COMMON/STOR5/TWT,NWT,WPR,WEIGHT,MASS
      COMMON/STOR8/OMEGA,RE,RA,RB,G0,GM,ALIM
      COMMON/STOR9/S,NK
      COMMON/STOR10/ST
      COMMON/STOR15/Q,VX,VY,VZ,VT,EL,AZ,RANGE,LAT,LONG,ACC,GDLAT
      COMMON/STOR17/LENGTH,SEL,DIST,LVEL,LACC
      COMMON/STOR19/TIME,TO
      COMMON/STOR20/AREA,ICPT,TIM,NTH,THS,THRUST
      COMMON/STOR25/LOW,EPTINY,EPBIG,PTI,PI,PI2,DT,HIGH
      DIMENSION ST(70,10)
      DIMENSION MCH(50),CDD(50),TWT(50),WPR(50),TIM(50),THS(50),S(9,6)
      REAL LVEL,LACC,LENGTH,LVEL2,LACC2,MASS
      EXTERNAL ACCEL2
      IF(LENGTH .LT. 0.1) RETURN
      CALL ATMSPH
      TIME = TO
      DT = AMIN1(0.01,DT)
      CALL TAB2
      CALL TAB3
      MASS = WEIGHT/G0
      LACC = (THRUST - WEIGHT*SEL)/MASS
      N=1
      S(1,N) = 0.
      S(4,N) = VT
      S(7,N) = LACC
      ST(1,1) = S(1,1)
      ST(4,1) = S(4,1)
      ST(7,1) = S(7,1)
  100 CONTINUE
      DIST2 = DIST
      ALT2 = ALT
      LVEL2 = LVEL
      TIME2 = TIME
      CALL RUNGE(ACCEL2)
      DT = AMIN1(0.01,DT)
      ALT = DIST*SEL
      IF(DIST .LT. LENGTH) GO TO 100
      TFRAC = (LENGTH - DIST2)/(DIST - DIST2)
      ALT = ALT2 + TFRAC*(ALT - ALT2)
      LVEL = LVEL2 + TFRAC*(LVEL - LVEL2)
      TIME = TIME2+ TFRAC*(TIME-TIME2)
      VT = LVEL
  900 RETURN
      END
```

```
CMAT        SUB. MAT --- ROTATION MATRIX AND INVERSE
      SUBROUTINE MAT
      DOUBLE PRECISION TMAT,SMAT
      DIMENSION TMAT(3,3), SMAT(3,3)
      COMMON/STOR3/TMAT,SMAT,SPHI,CPHI,CL,SL
      TMAT(1,1) = -SPHI*CL
      TMAT(1,2) = -SPHI*SL
      TMAT(1,3) = CPHI
      TMAT(2,1) = -SL
      TMAT(2,2) = CL
      TMAT(2,3) = 0.
      TMAT(3,1) = -CPHI*CL
      TMAT(3,2) = -CPHI*SL
      TMAT(3,3) = -SPHI
      SMAT(1,1) = TMAT(1,1)
      SMAT(1,2) = TMAT(2,1)
      SMAT(1,3) = TMAT(3,1)
      SMAT(2,1) = TMAT(1,2)
      SMAT(2,2) = TMAT(2,2)
      SMAT(2,3) = TMAT(3,2)
      SMAT(3,1) = TMAT(1,3)
      SMAT(3,2) = TMAT(2,3)
      SMAT(3,3) = TMAT(3,3)
      RETURN
      END
```

C.C.

```
CMATLCH     SUB. MATLCH --- LAUNCH SITE MATRIX
            SUBROUTINE MATLCH
            DOUBLE PRECISION TMAT,SMAT
            COMMON/STOR3/TMAT,SMAT,SPHI,CPHI,CL,SL
            COMMON/STOR8/OMEGA,RE,RA,RB,G0,GM,ALIM
            COMMON/STOR19/TIME,T0
            COMMON/STOR23/SPHI0,CPHI0,LAMDA0,X0,Y0,Z0
            DIMENSION TMAT(3,3),SMAT(3,3)
            REAL LBAR, LAMDA0
            SPHI = SPHI0
            CPHI = CPHI0
            LBAR = LAMDA0 + OMEGA*(TIME - T0)
            SL = SIN(LBAR)
            CL = COS(LBAR)
            CALL MAT
            RETURN
            END
```

```
CMATRAD     SUB. MATRAD --- RADAR SITE MATRIX
      SUBROUTINE MATRAD
      DOUBLE PRECISION TMAT,SMAT
      COMMON/STOR3/TMAT,SMAT,SPHI,CPHI,CL,SL
      COMMON/STOR8/OMEGA,RE,RA,RB,GO,GM,ALIM
      COMMON/STOR19/TIME,TO
      COMMON/STOR31/SPHI1,CPHI1,LAMDA1
      DIMENSION SMAT(3,3),TMAT(3,3)
      REAL LBAR,LAMDA1
      SPHI = SPHI1
      CPHI = CPHI1
      LBAR = LAMDA1 + OMEGA*(TIME - TO)
      SL = SIN(LBAR)
      CL = COS(LBAR)
      CALL MAT
      RETURN
      END
```

```
CMATROC     SUB. MATROC --- PRESENT POSITION MATRIX
      SUBROUTINE MATROC
      DOUBLE PRECISION TMAT,SMAT
      COMMON/STOR2/X,Y,Z,R
      COMMON/STOR3/TMAT,SMAT,SPHI,CPHI,CL,SL
      DIMENSION TMAT(3,3),SMAT(3,3)
      R = SQRT(X**2 + Y**2 + Z**2)
      SRT = SQRT(X**2 + Y**2)
      SPHI = Z/R
      CPHI = SRT/R
      CL = X/SRT
      SL = Y/SRT
   10 CONTINUE
      CALL MAT
      RETURN
      END
```

```
      COUT         SUB. OUT --- INTERPOLATE DATA AND PRINT
            SUBROUTINE OUT
            COMMON/STOR16/SA,ARRAY,NN
            COMMON/STOR18/TFRAC,LINE,PTIME,CONV,IEND,IPRINT,NPAGE,KPAG
            COMMON/STOR21/MESS,PHAS
            COMMON/STOR32/NSYS,CON
            COMMON/STOR33/NIIP,NCYC,NLIN
            COMMON/STOR34/KLIN
            DIMENSION CON(10,7),ARREY(10,30,10)
            DIMENSION MESS(30)
            DIMENSION NPAGE(10)
            DIMENSION SA(10,2,10),ARRAY(10,30,10),NVAR(06), TIMA(30)
            REAL MESS
         19 FORMAT(1H1)
         20 FORMAT(1X,A6,1X,F8.2,5X,F8.3,5X,F8.3,5X,F8.3,5X,F8.3,5X,F8.0,5X,
           1 F11.2,5X,F8.4,5X,F9.4)
         21 FORMAT(1X,A6,1X,F8.2,5X,F8.0,5X,F7.0,5X,F7.0,5X,F6.2,5X,F9.2,5X,
           1 F8.2,5X,F10.2,5X,F7.1)
         22 FORMAT(1X,A6,1X,F8.2,2X,F9.2,2X,F8.3,2X,F8.3,2X,F10.1,2X,
           1 F9.2,2X,F9.2,2X,F9.2,2X,F10.1,2X,F10.1,2X,F10.1)
         23 FORMAT(1X,A6,1X,F8.2,2X,F10.0,2X,F10.0,2X,F10.0,2X,F9.1,2X,F9.1,
           1 2X,F9.1,2X,F10.0,2X,F7.3,2X,F9.1,2X,F9.1)
         24 FORMAT(8X,8H  TIME   ,2X,10H    XL     ,2X,10H    YL    ,2X,
           1 10H    ZL    ,2X,9H   VXL    ,2X,9H   VYL    ,2X,9H   VZL    ,2X,
           2 10H    RXY   ,2X,7H  GAML ,2X,9H  VEL-L  ,2X,9H  RLDOT  )
         25 FORMAT(8X,8H  (SEC) ,2X,10H   (FT)    ,2X,10H   (FT)    ,2X,
           1 10H    (FT)   ,2X,9H (FT/SEC),2X,9H (FT/SEC),2X,9H (FT/SEC),2X,
           2 10H    (FT)   ,2X,7H (DEG) ,2X,9H (FT/SEC),2X,9H (FT/SEC)//)
         26 FORMAT(1X,A6,1X,F8.2,5X,F8.4,5X,F9.4,5X,F10.2,5X,F8.1)
         27 FORMAT(1X,A6,1X,F8.2,5X,F11.0,5X,F8.0,5X,F8.1)
         28 FORMAT(20X,47HINSTANTANEOUS IMPACT POINTS (VACUUM TRAJECTORY)//)
         30 FORMAT(8X,8H  TIME   ,5X,8H  ALPHA ,5X,8H  TH EL ,5X,8H  FL EL ,
           1 5X,8H  FL AZ ,5X,8H   ALT  ,5X,11H    RANGE   ,5X,8H LAT GD ,
           2 5X,9H   LONG  )
         31 FORMAT(8X,8H  (SEC) ,5X,8H  (DEG) ,5X,8H  (DEG) ,5X,8H  (DEG) ,
           1 5X,8H  (DEG) ,5X,8H   (FT) ,5X,11H   (N.M.)   ,5X,8H  (DEG) ,
           2 5X,9H   (DEG) //)
         32 FORMAT(8X,8H  TIME   ,5X,8H THRUST ,5X,7H WEIGHT,5X,7H  DRAG ,
           1 5X,6H MACH ,5X,9H  DYN PR ,5X,8H REL ACC,5X, 9H REL VEL ,5X,
           2 7H  MASS )
         33 FORMAT(8X,8H  (SEC) ,5X,8H  (LBS) ,5X,7H  (LBS),5X,7H  (LBS),
```

```
     1 11X,5X,9H(LBS/FT2),4X,9H(FT/SEC2),5X,10H (FT/SEC) ,5X,
     2 7H (SLUG)//)
  34 FORMAT(8X,8H  TIME  ,2X,9H SL RANGE,2X,8H LOOK AZ,2X,8H LOOK EL,
     1 2X,10H    VEL    ,2X,9H   EAST   ,2X,9H  NORTH  ,2X,9H   VERT   ,
     2 2X,10H   VEL-E   ,2X,10H   VEL-N   ,2X,10H   VEL-V   )
  35 FORMAT(8X,8H  (SEC) ,2X,9H  (N.M.) ,2X,8H  (DEG) ,2X,8H  (DEG) ,
     1 2X,10H  (FT/SEC),2X,9H   (N.M.),2X,9H  (N.M.) ,2X,9H  (N.M.) ,
     2 2X,10H (FT/SEC) ,2X,10H (FT/SEC) ,2X,10H (FT/SEC) //)
  36 FORMAT(20X,31H LAUNCH SITE COORDINATE SYSTEM //)
  37 FORMAT(20X,30H RADAR SITE COORDINATE SYSTEM //)
  38 FORMAT(20X,29H INERTIAL COORDINATE SYSTEM //)
  42 FORMAT(8X,8H  TIME  ,5X,8H LAT GD ,5X,9H   LONG   ,5X,10H   RANGE
     1,5X,8H IP TIME)
  43 FORMAT(8X,8H  (SEC) ,5X,8H  (DEG) ,5X,9H   (DEG) ,5X,10H   (N.M.)
     1,5X,8H (SEC)  //)
  44 FORMAT(8X,8H  TIME  ,5X,11H     R     ,5X,8H   VEL  ,5X,
     1 8H  ACCEL )
  45 FORMAT(8X,8H  (SEC) ,5X,11H    (FT)   ,5X,8H(FT/SEC),5X,
     1 9H(FT/SEC2)//)
  51 FORMAT(8X,8H  (SEC) ,5X,8H  (DEG) ,5X,8H  (DEG) ,5X,8H  (DEG) ,
     1 5X,8H  (DEG) ,5X,8H   (M)  ,5X,11H    (KM)   ,5X,8H  (DEG) ,
     2 5X,9H   (DEG) //)
  53 FORMAT(8X,8H  (SEC) ,5X,8H   (NT) ,5X,7H   (NT),5X,7H   (NT),
     1 11X,5X,9H (NT/M2) ,4X,9H (M/SEC2),5X,10H (M/SEC)   ,5X,
     2 7H   (KG) //)
  55 FORMAT(8X,8H  (SEC) ,2X,9H   (KM)  ,2X,8H  (DEG) ,2X,8H  (DEG) ,
     1 2X,10H   (M/SEC),2X,9H    (KM) ,2X,9H   (KM)  ,2X,9H   (KM)  ,
     2 2X,10H  (M/SEC) ,2X,10H  (M/SEC) ,2X,10H  (M/SEC) //)
  56 FORMAT(8X,8H  (SEC) ,2X,10H    (M)   ,2X,10H    (M)   ,2X,
     1 10H    (M)   ,2X,9H  (M/SEC),2X,9H  (M/SEC),2X,9H  (M/SEC),2X,
     2 10H    (M)   ,2X,7H (DEG) ,2X,9H  (M/SEC),2X,9H  (M/SEC)//)
  93 FORMAT(8X,8H  (SEC) ,5X,8H  (DEG) ,5X,9H   (DEG) ,5X,10H     (KM)
     1,5X,8H (SEC)  //)
  95 FORMAT(8X,8H  (SEC) ,5X,11H     (M)   ,5X,8H (M/SEC),5X,
     1 9H (M/SEC2)//)
 101 FORMAT( 120X,6H PAGE ,.4,1HA//)
 102 FORMAT( 120X,6H PAGE ,I4,1HB//)
 103 FORMAT( 120X,6H PAGE ,I4,1HC//)
 104 FORMAT( 120X,6H PAGE ,I4,1HD//)
 105 FORMAT( 120X,6H PAGE ,I4,1HE//)
```

```
  106 FORMAT( 120X,6H PAGE ,I4,1HF//)
      DATA BLNK/O377777777777/
      DATA BLKK/O20202020202O/
      DATA NVAR/8,8,10,10,4,3/
      IF(NIIP .EQ. 0) GO TO 300
      L = LINE + 1
      KLIN = L
      NLIN = NLIN + 1
      IF(NLIN .EQ. NCYC) GO TO 300
      GO TO 301
  300 IF(LINE .EQ. 30) LINF=0
      NLIN = 0
      LINE = LINE + 1
      L = LINE
      KLIN = L
      MESS(L) = PHAS
      PHAS = BLKK
  301 CONTINUE
      TIMA(L) = PTIME
      DO 11 J = 1,6
      NV = NVAR(J)
      DO 10 I = 1,NV
      ARRAY(I,L,J) = SA(I,2,J) + TFRAC*(SA(I,1,J) - SA(I,2,J))
      ARREY(I,L,J) = ARRAY(I,L,J)*CON(I,J)
      IF(SA(4,2,2) .GE. 0.0 .AND. SA(4,1,2) .GE. 0.0) GO TO 10
      ARRAY(4,L,2) = BLNK
      ARREY(4,L,2) = BLNK
   10 CONTINUE
   11 CONTINUE
C        CONVERT TO DEGREES
      ARRAY(1,L,1) = ARRAY(1,L,1)/CONV
      ARRAY(2,L,1) = ARRAY(2,L,1)/CONV
      ARRAY(3,L,1) = ARRAY(3,L,1)/CONV
      ARRAY(4,L,1) = ARRAY(4,L,1)/CONV
      ARRAY(7,L,1) = ARRAY(7,L,1)/CONV
      ARRAY(8,L,1) = ARRAY(8,L,1)/CONV
      ARRAY(8,L,3) = ARRAY(8,L,3)/CONV
      ARRAY(2,L,4) = ARRAY(2,L,4)/CONV
      ARRAY(3,L,4) = ARRAY(3,L,4)/CONV
      ARRAY(1,L,5) = ARRAY(1,L,5)/CONV
```

VXL, VZL	-	are velocity components in the downrange and vertical directions. Found by integration of AXL and AZL.
CS	=	COS(THETA)
THETA	-	is the inclination angle of the rocket, the angle between the principal body axis and the horizontal. It is originally set to the launch elevation angle, and is modified by double integration of THDD, THETA double dot.
FNORM	=	CNA*SAREA*Q*ALPH
CNA	-	is the slope of the coefficient of the normal force acting on the center of pressure.
SAREA	-	is the reference area.
Q	-	is defined above.
ALPH	-	THETA - GAMA, and is called the "angle of attack". The program switches from the Period I model to the model used for the rest of the flight when ALPH becomes zero (found by interpolation as it crosses from plus to minus). The duration of Period I is about 0.8 second for a NIKE-CAJUN, and about 5 seconds for a SCOUT.
GAMA	=	ATAN (VZL/VXL)
		GAMA is the flight path angle which defines the di[illegible] in which the rocket is moving
MASS	-	is the weight of the rocket divided by the force of gr[illegible]ity.
AZL	-	is acceleration in the Z_L direction, which is vertical at the launch site.
THDD	-	is THETA double dot, the acceleration in the rocket's inclination angle to the X_L axis.
LSM	=	LCP-LCG.
LCP	-	is the distance from the reference position (usually the nose of the rocket) to the center of pressure.
LCG	-	is the distance from the reference position to the rocket's center of gravity.
INERT	-	is the rocket's pitch moment of inertia.

The equations above have been simplified by the elimination of effects such as pitch damping and jet damping which were found to have no significant

```
      ARRAY(2,L,5) = ARRAY(2,L,5)/CONV
      ARRFY(1,L,1) = ARRFY(1,L,1)/CONV
      ARRFY(2,L,1) = ARRFY(2,L,1)/CONV
      ARRFY(3,L,1) = ARRFY(3,L,1)/CONV
      ARRFY(4,L,1) = ARRFY(4,L,1)/CONV
      ARRFY(7,L,1) = ARRFY(7,L,1)/CONV
      ARRFY(8,L,1) = ARRFY(8,L,1)/CONV
      ARRFY(8,L,3) = ARRFY(8,L,3)/CONV
      ARRFY(2,L,4) = ARRFY(2,L,4)/CONV
      ARRFY(3,L,4) = ARRFY(3,L,4)/CONV
      ARRFY(1,L,5) = ARRFY(1,L,5)/CONV
      ARRFY(2,L,5) = ARRFY(2,L,5)/CONV
      CALL TPOUT
C         IF LINE=30 OR IEND=1 OR IPRINT=1, PRINTOUT
      IF(LINE .EQ. 30 .OR. IEND .EQ. 1  .OR. IPRINT .EQ. 1) GO TO 50
      GO TO 200
   50 CONTINUE
      KPAG = KPAG + 1
      GO TO (151,152,151),NSYS
  151 CONTINUE
      I = 1
      NPAG = NPAGE(I)
      I = I + 1
      GO TO (61,62,63,64,65,66,67,68),NPAG
   61 CONTINUE
      PRINT 101,KPAG
      PRINT 30
      PRINT 31
C        PRINT PAGE A (ENGLISH)
      PRINT 20, ((MESS(LL),TIMA(LL),(ARRAY(I,LL,1),I=1,08)),LL=1,L)
      PRINT 19
      NPAG = NPAGE(I)
      I = I + 1
      GO TO (61,62,63,64,65,66,67,68),NPAG
   62 CONTINUE
      PRINT 102,KPAG
      PRINT 32
      PRINT 33
C        PRINT PAGE B (ENGLISH)
      PRINT 21, ((MESS(LL),TIMA(LL),(ARRAY(I,LL,2),I=1,08)),LL=1,L)
```

```
      PRINT 19
      NPAG = NPAGE(I)
      I = I + 1
      GO TO (61,62,63,64,65,66,67,68),NPAG
   63 CONTINUE
      PRINT 103,KPAG
      PRINT 36
      PRINT 24
      PRINT 25
C        PRINT PAGE C (ENGLISH)
      PRINT 23, ((MESS(LL),TIMA(LL),(ARRAY(I,LL,3),I=1,10)),LL=1,L)
      PRINT 19
      NPAG = NPAGE(I)
      I = I + 1
      GO TO (61,62,63,64,65,66,67,68),NPAG
   64 CONTINUE
      PRINT 104,KPAG
      PRINT 37
      PRINT 34
      PRINT 35
C        PRINT PAGE D (ENGLISH)
      PRINT 22, ((MESS(LL),TIMA(LL),(ARRAY(I,LL,4),I=1,10)),LL=1,L)
      PRINT 19
      NPAG = NPAGE(I)
      I = I + 1
      GO TO (61,62,63,64,65,66,67,68),NPAG
   65 CONTINUE
      PRINT 105,KPAG
      PRINT 28
      PRINT 42
      PRINT 43
C        PRINT PAGE E (ENGLISH)
      PRINT 26, ((MESS(LL),TIMA(LL),(ARRAY(I,LL,5),I=1,04)),LL=1,L)
      PRINT 19
      NPAG = NPAGE(I)
      I = I + 1
      GO TO (61,62,63,64,65,66,67,68),NPAG
   66 CONTINUE
      PRINT 106,KPAG
      PRINT 38
```

```
      PRINT 44
      PRINT 45
C        PRINT PAGE E (ENGLISH)
      PRINT 27, ((MESS(LL),TIMA(LL),(ARRAY(I,LL,6),I=1,03)),LL=1,L)
      PRINT 19
   67 CONTINUE
   68 CONTINUE
      GO TO (199,152,152),NSYS
  152 CONTINUE
      I = 1
      NPAG = NPAGE(I)
      I = I + 1
      GO TO (81,82,83,84,85,86,87,88),NPAG
   81 CONTINUE
      PRINT 101,KPAG
      PRINT 30
      PRINT 51
C        PRINT PAGE A (METRIC)
      PRINT 20, ((MESS(LL),TIMA(LL),(ARREY(I,LL,1),I=1,08)),LL=1,L)
      PRINT 19
      NPAG = NPAGE(I)
      I = I + 1
      GO TO (81,82,83,84,85,86,87,88),NPAG
   82 CONTINUE
      PRINT 102,KPAG
      PRINT 32
      PRINT 53
C        PRINT PAGE B (METRIC)
      PRINT 21, ((MESS(LL),TIMA(LL),(ARREY(I,LL,2),I=1,08)),LL=1,L)
      PRINT 19
      NPAG = NPAGE(I)
      I = I + 1
      GO TO (81,82,33,84,85,86,87,88),NPAG
   83 CONTINUE
      PRINT 103,KPAG
      PRINT 36
      PRINT 24
      PRINT 56
C        PRINT PAGE C (METRIC)
      PRINT 23, ((MESS(LL),TIMA(LL),(ARREY(I,LL,3),I=1,10)),LL=1,L)
```

```
      PRINT 19
      NPAG = NPAGE(I)
      I = I + 1
      GO TO (81,82,83,84,85,86,87,88),NPAG
   84 CONTINUE
      PRINT 104,KPAG
      PRINT 37
      PRINT 34
      PRINT 55
C        PRINT PAGE D (METRIC)
      PRINT 22, ((MESS(LL),TIMA(LL),(ARREY(I,LL,4),I=1,10)),LL=1,L)
      PRINT 19
      NPAG = NPAGE(I)
      I = I + 1
      GO TO (81,82,83,84,85,86,87,88),NPAG
   85 CONTINUE
      PRINT 105,KPAG
      PRINT 28
      PRINT 42
      PRINT 93
C        PRINT PAGE E (METRIC)
      PRINT 26, ((MESS(LL),TIMA(LL),(ARREY(I,LL,5),I=1,04)),LL=1,L)
      PRINT 19
      NPAG = NPAGE(I)
      I = I + 1
      GO TO (81,82,83,84,85,86,87,88),NPAG
   86 CONTINUE
      PRINT 106,KPAG
      PRINT 38
      PRINT 44
      PRINT 95
C        PRINT PAGE F (METRIC)
      PRINT 27, ((MESS(LL),TIMA(LL),(ARREY(I,LL,6),I=1,03)),LL=1,L)
      PRINT 19
   87 CONTINUE
   88 CONTINUE
  199 CONTINUE
      DO 75 J=1,30
      MESS(J) = BLKK
   75 CONTINUE
      IPRINT = 0
      LINE = 0
  200 RETURN
      END
```

```
  CRUNGE       SUB. RUNGF --- 4TH ORDER R-K INTEGRATION
          SUBROUTINE RUNGE(ACCEL)
          DOUBLE PRECISION ST
          DOUBLE PRECISION DP
          COMMON/STOR8/OMEGA,RE,RA,RB,GO,GM,ALIM
          COMMON/STOR9/S,NK
          COMMON/STOR10/ST
          COMMON/STOR19/TIME,TO
          COMMON/STOR25/LOW,EPTINY,EPBIG,PTI,PI,PI2,DT,HIGH
          DIMENSION S(9,6),ST(20,10),K(6,4)
          DIMENSION DP(6)
          REAL LOW,K
C         BEGIN RUNGE-KUTTA INTEGRATION
          N = 1
          I = 2
      350 CONTINUE
          TIMEO = TIME
          DO 400 J=1,6
          K(J,1) = DT*S(J+3,1)
          S(J,2) = S(J,1) + K(J,1)/2.
      400 CONTINUE
          TIME = TIMEO + DT/2.
          NK = 2
          CALL ACCEL
          DO 404 J=1,6
          K(J,2) = DT*S(J+3,2)
          S(J,3) = S(J,1) + K(J,2)/2.
      404 CONTINUE
          TIME = TIMEO + DT/2.
          NK = 3
          CALL ACCEL
          DO 408 J=1,6
          K(J,3) = DT*S(J+3,3)
          S(J,4) = S(J,1) + K(J,3)
      408 CONTINUE
          TIME = TIMEO + DT
          NK = 4
          CALL ACCEL
          DO 411 J = 1,6
          K(J,4) = DT*S(J+3,4)
          ST(J+10,I  ) = (K(J,1) + 2.*K(J,2) + 2.*K(J,3) + K(J,4))/6.
```

c.c.

```
  411 CONTINUE
C         STORE THE SOLUTION
      DO 490 L=1,6
      ST(L,I   ) = ST(L,N) + ST(L+10,I   )
  490 CONTINUE
C
C
  494 IF(I .EQ. 4) GO TO 600
      IF(I .EQ. 2) GO TO 495
      GO TO 497
  495 CONTINUE
      TIME = TIME - DT
      DT = DT/2.
      I = I + 1
      N = 1
      GO TO 350
  497 CONTINUE
      I = I + 1
      N = 3
      DO 498 J = 1,6
      S(J,1) = ST(J,3)
  498 CONTINUE
      NK =1
      CALL ACCEL
      GO TO 350
  600 CONTINUE
      DT = 2.*DT
      DO 602 J=4,6
      IF(DABS(ST(J,2)) .LT. 1.0D-7
     1 .OR. DABS(ST(J,4)) .LT. 1.0D-7) GO TO 601
      DP(J) = DABS(1.0 - ST(J,4)/ST(J,2))
      GO TO 602
  601 DP(J) = 0.0
  602 CONTINUE
      FRAC = DMAX1(DP(4),DP(5),DP(6))
C         COMPARE THE DT SOLUTION WITH THE DT/2 SOLUTION
      IF(FRAC .LT. EPTINY) GO TO 610
      IF(FRAC .GT. EPBIG) GO TO 620
      GO TO 630
  610 DT = 2.*DT
```

D.C.

```
      IF(DT .GT. HIGH) DT = HIGH
      GO TO 630
  620 IF(DT .LT. LOW) GO TO 630
      TIME = TIME - DT
      DT = DT/2.
      N = 1
      I = 2
      DO 622 J=1,9
      S(J,1) = ST(J,1)
  622 CONTINUE
      DO 621 L=1,6
      ST(L,2) = ST(L,3)
  621 CONTINUE
      GO TO 350
  630 CONTINUE
      IF(DT .LT. LOW) DT = LOW
      I = 2
      N = 1
      ST(10,N) = TIME
      DO 635 M=1,6
C        STORE THE FINAL SOLUTION
      ST(M,N) = ST(M,4) + (ST(M,4) - ST(M,2))/15.
  635 CONTINUE
      DO 650 J=1,6
      S(J,N) = ST(J,N)
  650 CONTINUE
      TIME = ST(10,N)
      NK = 1
C        COMPUTE NEXT ACCELERATION
      CALL ACCEL
      RETURN
      END
```

```
CSETSA      SUB. SETSA --- STORE OUTPUT TEMPORARILY
      SUBROUTINE SETSA
      COMMON/STOR1/ALT,TEMP,PRES,DENS,VISC,SOUND
      COMMON/STOR2/X,Y,Z,R
      COMMON/STOR4/SAREA,MCH,CDD,CD,MACH,NCD,CDS,DRAG
      COMMON/STOR5/TWT,NWT,WPR,WEIGHT,MASS
      COMMON/STOR7/XDOT,YDOT,ZDOT,XDDOT,YDDOT,ZDDOT
      COMMON/STOR12/YL,VYL,RXY,GAML,RLDOT
      COMMON/STOR13/XL,ZL,THETA,VXL,VZL,ALPH,GAMA,CAZ,SAZ
      COMMON/STOR15/Q,VX,VY,VZ,VT,EL,AZ,RANGE,LAT,LONG,ACC,GDLAT
      COMMON/STOR16/SA,ARRAY,NN
      COMMON/STOR19/TIME,TO
      COMMON/STOR20/AREA,IOPT,TIM,NTH,THS,THRUST
      COMMON/STOR27/IIPLAT,IIPLON,IIPP,IIPTIM
      COMMON/STOR28/RLCH,AZLCH,ELLCH,VELLCH,ELCH,NLCH,ZLCH,EDLCH,
     1 NDLCH,ZDLCH
      COMMON/STOR29/RRAD,AZRAD,ELRAD,VELRAD,ERAD,NRAD,ZRAD,EDRAD,
     1 NDRAD,ZDRAD
      COMMON/STOR30/DELTA,VEL,ACCI
      REAL IIPLAT,IIPLON,IIPP,IIPTIM
      REAL NRAD,NLCH,NDRAD,NDLCH
      REAL LAMDA0,MASS ,LOW,LONG, LAT, MACH,MCH
      DIMENSION TIM(50),THS(50),TWT(50),WPR(50),MCH(50),CDD(50)
      DIMENSION SA(10,2,10),ARRAY(10,30,10)
      REAL MASS
      N = NN
      J = 1
      SA(1,N,J) = ALPH
      SA(2,N,J) = THETA
      SA(3,N,J) = EL
      SA(4,N,J) = AZ
      SA(5,N,J) = ALT
      SA(6,N,J) = RANGE/6076.1155
      SA(7,N,J) = GDLAT
      SA(8,N,J) = LONG
      J = 2
      SA(1,N,J) = THRUST
      SA(2,N,J) = WEIGHT
      SA(3,N,J) = DRAG
      SA(4,N,J) = MACH
      SA(5,N,J) = Q
```

```
      SA(6,N,J) = ACC
      SA(7,N,J) = VT
      SA(8,N,J) = MASS
      J = 3
      SA(1,N,J) = XL
      SA(2,N,J) = YL
      SA(3,N,J) = ZL
      SA(4,N,J) = VXL
      SA(5,N,J) = VYL
      SA(6,N,J) = VZL
      SA(7,N,J) = RXY
      SA(8,N,J) = GAML
      SA(9,N,J) = VELLCH
      SA(10,N,J) = RLDOT
      J = 4
      SA(1,N,J) = RRAD/6076.1155
      SA(2,N,J) = AZRAD
      SA(3,N,J) = ELRAD
      SA(4,N,J) = VELRAD
      SA(5,N,J) = ERAD/6076.1155
      SA(6,N,J) = NRAD/6076.1155
      SA(7,N,J) = ZRAD/6076.1155
      SA(8,N,J) = EDRAD
      SA(9,N,J) = NDRAD
      SA(10,N,J) = ZDRAD
      J = 5
      SA(1,N,J) = IIPLAT
      SA(2,N,J) = IIPLON
      SA(3,N,J) = IIPR/6076.1155
      SA(4,N,J) = IIPTIM
      J = 6
      SA(1,N,J) = R
      SA(2,N,J) = VEL
      SA(3,N,J) = ACCI
      RETURN
      END
```

```
CTABLE        SUB. TABLE --- THRUST,WEIGHT,DRAG INPUT
      SUBROUTINE TABLE
      COMMON/STOR4/SAREA,MCH,CDD,CD,MACH,NCD,CDS,DRAG
      COMMON/STOR5/TWT,NWT,WPR,WEIGHT
      COMMON/STOR6/P0,WRM,WPL,WPP2,WTM
      COMMON/STOR20/AREA,IOPT,TIM,NTH,THS,THRUST
      COMMON/STOR21/MESS,PHAS
      COMMON/STOR24/PHT,NID,NTAB,NPH,KPH
   10 FORMAT(1X,A6)
      DIMENSION MESS(30)
      DIMENSION PHT(14),NID(14),WTM(6),TSEP(6),WTP(6)
      DIMENSION TIM(50),THS(50),TWT(50),WPR(50),MCH(50),CDD(50)
      REAL LAMDA,MASS,K,LOW,LONG, LAT, MACH,MCH,LAMDA0
      NAMELIST /NAM2/AREA,TIM,NTH,THS,IOPT
      NAMELIST /NAM3/TWT,NWT,WPR
      NAMELIST /NAM4/SAREA,MCH,NCD,CDD
      IF(NPH .GT. NTAB) GO TO 900
      NNNN = NID(KPH)
      GO TO (343,344,345,349),NNNN
  343 READ(5,10) PHAS
      GO TO 349
  344 CONTINUE
C         READ IN DRAG TABLE
      READ(5,10) PHAS
      READ(5,NAM4)
      GO TO 349
  345 CONTINUE
C         READ IN THRUST, WEIGHT, AND DRAG TABLES
      READ(5,10) PHAS
      READ(5,NAM2)
      DO 346 J=1,NTH
      TIM(J) = TIM(J) + PHT(NPH)
  346 CONTINUE
      READ(5,NAM3)
      DO 347 J=1,NWT
      TWT(J) = TWT(J) + PHT(NPH)
  347 CONTINUE
      WPP2 = WPP2 - WPR(1)
      READ(5,NAM4)
      GO TO 349
  349 CONTINUE
  900 RETURN
      END
```

```
CTAB1          SUB. TAB1 --- DRAG COEFFICIENT
      SUBROUTINE TAB1
      COMMON/STOR4/SAREA,MCH,CDD,CD,MACH,NCD,CDS,DRAG
      DIMENSION MCH(50),CDD(50)
      REAL MACH,MCH
      IF(MACH .LE. MCH(1)) GO TO 116
      IF(MACH .GE. MCH(NCD)) GO TO 115
      DO 100 J=2,NCD
      IF(MACH .LT. MCH(J)) GO TO 110
  100 CONTINUE
  110 FRAC = (MACH - MCH(J-1))/(MCH(J) - MCH(J-1))
      CD = CDD(J-1) + FRAC*(CDD(J) - CDD(J-1))
      GO TO 200
  115 CD = CDD(NCD)
      GO TO 200
  116 CD = CDD(1)
  200 CDS = CD*SAREA
      RETURN
      END
```

```
      CTAB2          SUB. TAB2 --- THRUST INTERPOLATION
            SUBROUTINE TAB2
            COMMON/STOR1/ALT,TEMP,PRES,DENS,VISC,SOUND
            COMMON/STOR6/PO,WRM,WPL,WPP2,WTM
            COMMON/STOR19/TIME,TO
            COMMON/STOR20/AREA,IOPT,TIM,NTH,THS,THRUST
            DIMENSION TIM(50),THS(50),WTM(6)
            IF(TIME .LT. TIM(1)) GO TO 115
            IF(TIME .GE. TIM(NTH)) GO TO 115
            DO 100 J=2,NTH
            IF(TIME .LT. TIM(J)) GO TO 110
        100 CONTINUE
        110 TFRAC = (TIME - TIM(J-1))/(TIM(J)-TIM(J-1))
            THREF = THS(J-1) + TFRAC*(THS(J) - THS(J-1))
            GO TO (210,211),IOPT
        210 THRUST = THREF + AREA*(PO - PRES)
            GO TO 900
        211 THRUST = THREF - AREA*PRES
            GO TO 900
        115 THRUST = 0.0
        900 RETURN
            END
```

```
      CTAB3         SUB. TAB3 --- WEIGHT INTERPOLATION
            SUBROUTINE TAB3
            COMMON/STOR5/TWT,NWT,WPR,WEIGHT
            COMMON/STOR6/PO, WRM,WPL,WPP2,WTM
            COMMON/STOR19/TIME,TO
            DIMENSION TWT(50),WPR(50),WTM(6)
            IF(TIME .GE.TWT(NWT)) GO TO 115
            DO 100 J=2,NWT
            IF(TIME .LT. TWT(J)) GO TO 110
        100 CONTINUE
        110 TFRAC = (TIME - TWT(J-1))/(TWT(J) - TWT(J-1))
            WPP = WPR(J-1) + TFRAC*(WPR(J) - WPR(J-1))
            GO TO 116
        115 WPP = WPR(NWT)
        116 WEIGHT = WPP + WPP2 + WRM + WPL
        200 RETURN
            END
```

```
CTPOUT        SUR. TPOUT --- SPECIAL TAPE OUTPUT
      SUBROUTINE TPOUT
      COMMON/STOR16/SA,ARRAY,NN
      COMMON/STOR18/TFRAC,LINE,PTIME,CONV,IEND,IPRINT,NPAGE,KPAG
      COMMON/STOR34/KLIN
      DIMENSION TPA(75),SA(10,2,10),ARRAY(10,30,10),NPAGE(10)
      L = KLIN
      TPA( 1) = PTIME
      TPA( 2) = ARRAY(2,L,2)
      TPA( 3) = ARRAY(1,L,2)
      TPA( 4) = ARRAY(3,L,6)
      TPA( 5) = ARRAY(1,L,6)
      TPA( 6) = ARRAY(5,L,1)
      TPA( 7) = ARRAY(6,L,1)
      TPA( 8) = ARRAY(4,L,2)
      TPA( 9) = ARRAY(5,L,2)
      TPA(10) = ARRAY(7,L,1)
      TPA(12) = ARRAY(8,L,1)
      TPA(14) = ARRAY(1,L,1)
      TPA(18) = ARRAY(3,L,1)
      TPA(20) = ARRAY(4,L,1)
      TPA(23) = ARRAY(2,L,1)
      TPA(37) = ARRAY(7,L,2)
      TPA(55) = ARRAY(1,L,3)
      TPA(56) = ARRAY(2,L,3)
      TPA(57) = ARRAY(3,L,3)
      TPA(58) = ARRAY(4,L,3)
      TPA(59) = ARRAY(5,L,3)
      TPA(60) = ARRAY(6,L,3)
      TPA(61) = ARRAY(7,L,3)
      TPA(62) = ARRAY(8,L,3)
      TPA(63) = ARRAY(9,L,3)
      TPA(64) = ARRAY(10,L,3)
      TPA(65) = ARRAY(4,L,5)
      TPA(66) = ARRAY(1,L,5)
      TPA(67) = ARRAY(2,L,5)
      TPA(68) = ARRAY(3,L,5)
      WRITE(11) (TPA(J),J=1,71)
      DO 100 J=1,75
      TPA(J) = 0.0
  100 CONTINUE
      RETURN
      END
```

```
CTRANS1     SUB. TRANS1 --- INERTIAL TO ROTATING SYS.
      SUBROUTINE TRANS1(EX,EY,EZ,WX,WY,WZ)
      DOUBLE PRECISION TMAT,SMAT
      COMMON/STOR3/TMAT,SMAT,SPHI,CPHI,CL,SL
      DIMENSION TMAT(3,3),SMAT(3,3)
      WX = TMAT(1,1)*EX + TMAT(1,2)*EY + TMAT(1,3)*EZ
      WY = TMAT(2,1)*EX + TMAT(2,2)*EY + TMAT(2,3)*EZ
      WZ = TMAT(3,1)*EX + TMAT(3,2)*EY + TMAT(3,3)*EZ
      RETURN
      END
```

```
CTRANS2     SUB. TRANS2 --- ROTATING TO INERTIAL SYS.
      SUBROUTINE TRANS2(BX,BY,BZ,UX,UY,UZ)
      DOUBLE PRECISION TMAT,SMAT
      COMMON/STOR3/TMAT,SMAT,SPHI,CPHI,CL,SL
      DIMENSION TMAT(3,3),SMAT(3,3)
      UX = SMAT(1,1)*BX + SMAT(1,2)*BY + SMAT(1,3)*BZ
      UY = SMAT(2,1)*BX + SMAT(2,2)*BY + SMAT(2,3)*BZ
      UZ = SMAT(3,1)*BX + SMAT(3,2)*BY + SMAT(3,3)*BZ
      RETURN
      END
```